Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01944 / 2013
Datum: 3.9.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **NIKA ŽAGAR**

Naslov: **APLIKACIJA ZA VAREN PRENOS DATOTEK IN DIREKTORIJEV MED ODJEMALCEM IN STREŽNIKOM**
APPLICATION FOR SECURE TRANSFER OF FILES AND DIRECTORIES BETWEEN CLIENT AND SERVER

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Prenos datotek med odjemalcem in strežnikom mora imeti vsaj naslednje lastnosti: mora biti varen, zanesljiv in mora omogočati nadaljevanje prenova prekinjenih prenosov. Zasnujte, izdelajte načrt in razvijte aplikacijo, ki omogoča varen prenos datotek in direktorijskih struktur med odjemalcem in strežnikom. Pri tem naj se za kontrolo celotne vsebine prenesenih datotek uporablja zgoščevalni algoritem. Razvijte tudi mehanizme za nadaljevanje prenosa prekinjenih prenosov od točke prekinitve prenosa.

Mentor:
doc. dr. Rok Rupnik

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nika Žagar

Aplikacija za varen prenos datotek in direktorijev med odjemalcem in strežnikom

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Rok Rupnik

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

[1] V dogovorju z mentorjem lahko kandidat diplomsko delo s pripadajočo izvorno kodo izda tudi pod katero izmed alternativnih licenc, ki ponuja določen del pravic vsem: npr. Creative Commons, GNU GPL.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01944 / 2013
Datum: 3.9.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **NIKA ŽAGAR**

Naslov: **APLIKACIJA ZA VAREN PRENOS DATOTEK IN DIREKTORIJEV MED ODJEMALCEM IN STREŽNIKOM**
APPLICATION FOR SECURE TRANSFER OF FILES AND DIRECTORIES BETWEEN CLIENT AND SERVER

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Prenos datotek med odjemalcem in strežnikom mora imeti vsaj naslednje lastnosti: mora biti varen, zanesljiv in mora omogočati nadaljevanje prenova prekinjenih prenosov. Zasnujte, izdelajte načrt in razvijte aplikacijo, ki omogoča varen prenos datotek in direktorijskih struktur med odjemalcem in strežnikom. Pri tem naj se za kontrolo celotne vsebine prenesenih datotek uporablja zgoščevalni algoritem. Razvijte tudi mehanizme za nadaljevanje prenosa prekinjenih prenosov od točke prekinitve prenosa.

Mentor:

doc. dr. Rok Rupnik

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Nika Žagar, z vpisno številko **63080120**, sem avtorica diplomskega dela z naslovom:

Aplikacija za varen prenos datotek in direktorijev med odjemalcem in strežnikom

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom doc. dr. Roka Rupnika,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 15. oktobra 2013 Podpis avtorja:

Zahvaljujem se svoji družini, prijateljem ter sošolcem za vso podporo pri študiju in izdelavi diplomskega dela.

Moji mami.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Opredelitev problema 1

1.2 Namen in cilji 1

2 Uporabljene tehnologije **3**

2.1 ASP .NET MVC 4.0 3

2.1.1 Model 4

2.1.2 Pogled 4

2.1.3 Nadzornik 5

2.2 ASP .NET HTTP Handler 5

2.3 Spletni strežnik IIS 7

2.4 Servis Windows 8

2.5 ADO .NET Entity Framework 9

3 Zajem zahtev **11**

3.1 Strežnik 11

3.1.1 Spletna stran za administracijo strežnika 11

3.1.2 Spletna storitev 12

3.2 Odjemalec 17

3.2.1 Servisni modul 17

3.2.2 Uporabniški vmesnik za konfiguracijo servisnega modula 18

4 Analiza in načrtovanje 23

4.1 Konceptualni model 23
4.2 Logični model 25
4.2.1 Entiteta Users 25
4.2.2 Entiteta StorageRoot 26
4.2.3 Entiteta UserStorageRoot 26
4.2.4 Entiteta EventLogTransfer 26
4.2.5 Entiteta ActionType 27
4.2.6 Entiteta Path 28
4.2.7 Entiteta Roles 28
4.2.8 Entiteta UserRoles 28
4.3 Razredni diagram - spletna stran za administracijo strežnika . 29
4.4 Diagram zaporedja 30
4.4.1 Diagrami zaporedja za administracijo strežnika 30
4.4.2 Diagrami zaporedja spletnih storitev 31
4.5 Diagram postavitve 31

5 Implementacija 39

5.1 Implementacija spletne strani za administracijo strežnika . . . 39
5.2 Implementacija spletne storitve za izmenjavo datotek 42
5.3 Implementacija odjemalca 44
5.3.1 Servisni modul 44
5.3.2 Uporabniški vmesnik 45

6 Testiranje 49

7 Nadaljnji razvoj 53

8 Sklepne ugotovitve 55

Povzetek

Diplomsko delo zajema opis celotnega postopka razvoja programske rešitve, ki je namenjena enostavni izmenjavi datotek preko varne povezave s pomočjo protokola HTTP (SSL). Gre za aplikacijo tipa odjemalec - strežnik. Strežnik je sestavljen iz spletne storitve za izmenjavo datotek in spletne strani za administracijo strežnika. Odjemalec je realiziran kot servisni modul.

Programska rešitev deluje na principu inicializacije seje za prenos datotek s strani odjemalca. Prenos datotek in direktorijskih struktur se vedno izvede z vnaprej definiranimi pravili. Pri izmenjavi se uporablja mehanizem kontrole prenosa celotne vsebine s pomočjo zgoščevalnih algoritmov. Rešitev poleg kontrole omogoča prav tako nadaljevanje prekinjenih prenosov.

V prvem delu diplomske naloge so opisane tehnologije in orodja, ki so bile uporabljene pri razvoju produkta. Sledi podroben opis zajema zahtev. V poglavju o analizi in načrtovanju se seznanimo s podatkovnim in statičnim vidikom načrtovanja sistema. Zajet je tudi podroben opis prenosa datotek in nekaterih funkcionalnosti spletne strani za administracijo strežnika. Na koncu sledi še kratek povzetek testiranja in ideje za nadaljnji razvoj produkta.

Ključne besede

prenos datotek, spletna storitev, generični upravljavec, servis Windows, MVC

Abstract

The thesis describes the development of a client-server application for simple data transfer via a secure connection using the HTTP (SSL) protocol. Server part is composed of a web service used for data transfer and a website used for administration purposes while the client is realized as a service module.

Data transfer session is initiated from the client side. The transfer of files and folder structures is executed based on the predefined set of rules. Transfer is controlled and verified by hash algorithms. Optimized continuation of an aborted transfer is also supported.

The thesis starts with a description of the technologies and tools that were used in the development of the solution. Following is a detailed description of the requirements. Static view of project planning and data analysis is presented. An explanation of the file transfer, some of the server administration website features and development specifics is also included. The thesis is concluded with a brief summary about testing and further development options and ideas.

Keywords

data transfer, web service, generic handler, Windows service, MVC

Poglavje 1

Uvod

1.1 Opredelitev problema

V podjetju se vsakodnevno prenaša velika količina datotek. Podjetje je že imelo podoben produkt, ki pa ni omogočal dopolnitve z novimi zahtevami. Posledično je bilo treba implementirati nov produkt, ki je skladen z zahtevami.

1.2 Namen in cilji

Namen diplomske naloge je razvoj in izdelava programske opreme, ki bo omogočala varno in učinkovito izmenjavo datotek. Za modeliranje smo uporabili jezik UML (ang. Unified Modeling Language). Pri razvoju smo uporabili programski jezik C# .NET, podatkovni strežnik MS SQL, spletni strežnik IIS ter orodje za modeliranje Power Designer.

Cilji naloge so:

- razviti produkt, ki bo izvajal učinkovito izmenjavo datotek preko varne povezave med dvema lokacijama,
- spoznati celoten postopek razvoja produkta od analize do testiranja skladno z metodologijo razvoja podjetja in

- razviti produkt, ki bo temeljil na modernih tehnologijah.

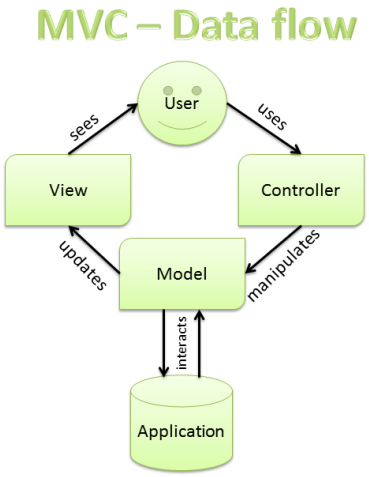
Poglavje 2

Uporabljene tehnologije

Zahteve za izgradnjo produkta so bile, da mora biti rešitev razvita v ogrodju ASP .NET MVC 4.0. Spletna storitev za komunikacijo med odjemalcem in strežnikom pa preko HTTP-protokola z uporabo metod GET in POST. Podatkovna zbirka, ki jo strežnik uporablja, je Microsoft SQL Server 2008 R2. Strežnik prav tako ne deluje kot samostojna storitev, ampak v okviru Microsoft Internet Information Server (IIS). Odjemalec je servisni modul v okolju Microsoft Windows. Za razvoj spletne storitve je bila uporabljena tehnologija ASP .NET Generic Handler.

2.1 ASP .NET MVC 4.0

ASP .NET MVC 4.0 je ogrodje za izgradnjo spletnih aplikacij. MVC je arhitekturni vzorec, ki razbije aplikacijo na tri glavne komponente. To so model (model), pogled (view) in nadzornik (controller). Ogrodje je izredno lahek reprezentacijski vzorec, ki je integriran v že obstoječe funkcije ASP .NET. Z vmesno komponento "nadzornik" je mogoče ločiti podatke in poslovno logiko od uporabniškega vmesnika. V nadaljevanju so v podpoglavjih razložene tri glavne komponente ogrodja. [10]



Slika 2.1: Grafični prikaz MVC-komponent [15]

2.1.1 Model

Model je komponenta, ki predstavlja poslovni del aplikacije in implementira logiko aplikacijske podatkovne domene. Objekti modela pridobijo in shranijo stanje modela v podatkovno bazo. Na primer: imamo objekt oseba, njegove informacije se pridobijo iz podatkovne baze. Nad temi informacijami se izvedejo operacije. Posodobljene informacije se nato zapišejo nazaj v podatkovno bazo. [10]

2.1.2 Pogled

Pogledi so komponente, ki prikazujejo uporabniške vmesnike. Tipično je uporabniški vmesnik ustvarjen na podlagi podatkov, pridobljenih iz modela. Kot primer lahko navedemo pogled, ki bi omogočal urejanje objekta osebe. [10]

2.1.3 Nadzornik

Nadzorniki so komponente, ki nadzorujejo uporabnikovo interakcijo, delo z modelom in prikazujejo poglede. V MVC-aplikaciji pogled le prikaže informacije, nadzornik pa nadzoruje in odgovarja na uporabnikove vhode in interakcije. Na primer: nadzornik upravlja poizvedbe, ki jih nato posreduje modelu. [10]

2.2 ASP .NET HTTP Handler

ASP .NET Handler (HTTP-upravljavec) je proces, ki se izvaja kot odgovor na zahtevo, ki je poslana spletni aplikaciji ASP .NET. Najbolj pogosti upravljavci so t. i. upravljavci strani (page handler), ki procesirajo datoteke .aspx. Ko uporabnik pošlje zahtevo za stran .aspx, se zahteva sprocesira preko t. i. upravljavca strani. Ampak to je le eden izmed tipov upravljavcev. ASP .NET ima tudi ostale vgrajene upravljavce, kot so na primer upravljavci za datoteke .asmx.

HTTP-upravljavci imajo dostop do vsebine in stanja aplikacije, informacije o seji in identitete uporabnika, ki pošilja zahtevo. HTTP-upravljavci so lahko asinhroni ali sinhroni. Sinhroni upravljavec ne pošlje odgovora, dokler prejeta zahteva ni sprocesirana. Asinhroni upravljavec pa neodvisno sprocesira zahtevo in pošlje odgovor.

ASP .NET mapira HTTP-zahteve skladno s končnicami datotek. ASP. NET vsebuje kar nekaj vgrajenih HTTP-upravljavcev, ki so predstavljeni v tabeli 2.1. [9]

Vsak razred, ki implementira System.Web.IHttpHandler, je lahko tarča HTTP-zahtev. Vmesnik IHttpHandler je sestavljen iz ProcessRequest in IsReusable. Podrobnosti vmesnika so predstavljene v tabeli 2.2.

Upravljavec pridobi objekt HttpContext, ki se prenese do metode ProcessRequest. HttpContext pridobi vse potrebne informacije o zahtevi preko objekta HttpRequest. Odgovor je zgeneriran preko objekta HttpReponse. Pri generiranju odgovora upravljavec nima nobenih omejitev.

Handler	Opis
`ASP .NET Page Handler`	Privzeti HTTP-upravljavec za vse strani ASP .NET
`Web service handler`	Privzeti HTTP-upravljavec za spletne storitve ASP .NET
`ASP .NET user control handler`	Privzeti HTTP-upravljavec za uporabniški krminilnik
`Trace handler`	Upravljavec, ki prikazuje trenutne sledi strani

Tabela 2.1: Vgrajeni HTTP-upravljavci

Ime	Opis
`ProcessRequest (context)`	Metoda se kliče, ko ogrodje ASP .NET hoče, da upravljavec zgenerira odgovor na zahtevo. Parameter je objekt HTTPContext, ki zagotavlja dostop do vseh podrobnosti zahteve
`IsReusable`	Lastnost, ki pove, ali lahko eno instanco HTTP-upravljavca uporabimo za procesiranje več zahtev.

Tabela 2.2: Metode in spremenljivke vmesnika IHttpHandler

Poznamo dva načina kreiranja upravljavcev: generični upravljavec ("generic handler") in upravljavec po meri ("custom handler"). V diplomski nalogi je bil uporabljen generični upravljavec. Generični upravljavci so hitri in preprosti za izdelavo. Do generičnih upravljavcev lahko dostopamo le preko URL-naslova. Če želimo, da se upravljavec odzove glede na tip http-zahteve ali tip url-naslova, potem moramo uporabiti upravljavce po meri. To dosežemo s konfiguracijo datoteke Web.config. [2]

S postavitvijo spletne storitve s pomočjo generičnega upravljavca se lahko izogneš skrivnosti "črni škatli" Microsofta, na katero naletiš pri uporabi standardne ASMX-tehnologije za pošiljanje in sprejemanje http-sporočil. Z generičnim upravljavcem pridobimo popoln nadzor nad prejetimi zahtevami in generiranjem odgovorov. S tem pridobimo spletno storitev, ki je zelo prilagodljiva, razširljiva in zelo enostavna za vzdrževanje.

2.3 Spletni strežnik IIS

Microsoftov spletni strežnik (IIS) je paleta internetnih procesov za strežnike z operacijskim sistemom Microsoft Windows. IIS podpira protokole, kot so FTP (File Transfer Protocol), NNTP (Network News Transfer Protocol) in SMTP (Simple Mail Transfer Protocol). Odlično je integriran z aplikacijami Microsoft .NET. [3]

Je tretji najbolj razširjen strežnik, takoj za Apache HTTP Server in nginx. IIS je bil vse do leta 2011 drugi najpopularnejši strežnik. IIS je sestavni del družine Windows Server. Prav tako je sestavni del nekaterih verzij Windows XP, Windows Vista, Windows 7 in 8. IIS ni omogočen takoj ob namestitvi operacijskega sistema, vendar ga lahko omogočimo preko nadzorne plošče. [13]

Prve verzije IIS so imele veliko ranljivosti in so bile vzrok velikemu številu vdorov v strežnike Windows. V različicah do verzije 5.1. so se vsi procesi izvajali pod sistemskim računom. Z verzijo 6.0 pa se je to spremenilo in vsi procesi tečejo pod računom Network Services. To pomeni, da če pride do

izkoriščanja programske kode, delovnim procesom ne bo dovoljen dostop do celotnega sistema. [13]

IIS vsebuje več komponent, ki opravljajo pomembne funkcije za aplikacijo in vloge spletnega strežnika. Vsaka komponenta ima svoje zadolžitve, kot so poslušanje zahtev, ki pridejo do strežnika, upravljanje procesov in branje konfiguracijskih datotek.

IIS ima modularno arhitekturo. Za razliko od enovitnega strežnika, pri katerem tečejo vsi procesi, ima IIS core web server engine, kar pomeni, da se izvajajo le procesi, ki jih zahtevamo. Z uporabo dodatnih modulov lahko razširimo funkcionalnosti strežnika. Na voljo so naslednji moduli:

- moduli HTTP,
- varnostni moduli,
- vsebinski moduli,
- kompresijski moduli,
- pomnilniški moduli,
- dnevniški in diagnostični moduli.

[13]

2.4 Servis Windows

Na operacijskih sistemih Windows je aplikacija servis Windows, ki teče v ozadju in opravlja specifične funkcije. Zasnovana je tako, da ne potrebuje interakcije uporabnika. Servisi Windows so lahko skonfigurirani tako, da se zaženejo, ko se prižge računalnik, in tečejo v ozadju toliko časa, dokler je računalnik prižgan, ali pa jih zaženemo takrat, ko je to potrebno. Koncept servisov Windows je podoben konceptu demonov Unix. Servisi Windows so

idealni za uporabo na serverjih ali kadar želimo implementirati dolgo trajajočo funkcionalnost, pri kateri ni zaželena uporabnikova interakcija. Prav tako servis lahko teče pod poljubnim uporabniškim računom. [14]

Najbolj znani servisi so:

- Active Directory Service,
- Event log,
- Windows update,
- Task Scheduler in
- DNS Client.

2.5 ADO .NET Entity Framework

ADO .NET Entity Framework je ogrodje za ustvarjanje entitetnega podatkovnega modela. V osnovi omogoča aplikacijam dostop do podatkov in spreminjanje teh. Ti podatki so predstavljeni kot entitete in razmerja v konceptualnem modelu. [11]

Arhitektura za dostop do podatkov je sestavljena iz štirih glavnih komponent, ki so: [4]

- objektne storitve,
- ponudniki podatkov na nivoju entitet,
- ponudniki podatkov ADO. NET in
- entitetni podatkovni model.

Objektne storitve omogočajo ustvarjanje, spreminjanje in shranjevanje objektov v podatkovno bazo. Postavljene so čisto na vrh ogrodja ter nudijo funkcionalnosti, ki jo uporablja največ razvijalcev.

Ponudnik podatkov na nivoju entitet ali EntityClient je ponudnik podatkov (podobno kot SqlClient), le na višjem nivoju. Deluje na nivoju entitnega

podatkovnega modela. Ponudnik podatkov je zmožen izvajati povpraševanja. Razlika med tem in objektnimi storitvami je ta, da rezultate, ki jih vrača ponudnik podatkov, lahko samo beremo, medtem ko objektne storitve omogočajo operacije sprememb in shranjevanja v podatkovno bazo. [8] [1]

Komponenta ponudnik podatkov ADO .NET omogoča dostop do posameznih podatkovnih baz. Vsak proizvajalec podatkovne baze implementira svojega t. i. ponudnika, ki zajema specifikacije dostopa do svoje podatkovne baze. [11]

Entitetni podatkovni model (EDM ali Entity data model) je skupek konceptov, ki opisujejo strukturo podatkov. Omogoča višji nivo abstrakcije, saj se vse dogaja na konceptualnem modelu in je logično ločen od tabel na podatkovni bazi. [16]

Prednost modela je v načinu načrtovanja informacijskega sistema. Poznamo tri možnosti, in sicer:

- Database first,
- Model first in
- Meet in the middle (kombinacija obeh zgoraj naštetih).

V mojem primeru je bil uporabljen koncept Model first.

Model first ali najprej model je pristop, ki omogoča, da ustvarimo svojo shemo podatkovne baze, ki temelji na modelu. Razvoj se začne brez definirane podatkovne baze. Najprej se ustvari konceptualni model, nato se uporabi orodje za generiranje podatkovnih baz ("Generate database wizard").

Poglavje 3

Zajem zahtev

Primarna naloga produkta je učinkovita izmenjava datotek preko varne povezave med dvema lokacijama. Gre za aplikacijo tipa strežnik - odjemalec. Programska rešitev je sestavljena iz naslednjih sklopov:

- podatkovne baze,
- strežnika za izmenjavo podatkov,
    1. spletne strani za administracijo strežnika,
    2. integracijskega vmesnika za izmenjavo podatkov,
- odjemalca za izmenjavo podatkov,
    1. servisnega modula in
    2. uporabniškega vmesnika za konfiguracijo servisnega modula.

Zajem zahtev je tako razdeljen na dva dela, to sta strežnik in odjemalec.

3.1 Strežnik

3.1.1 Spletna stran za administracijo strežnika

Zaradi večje preglednosti so na sliki diagrama primerov uporabe 3.1 nekateri primeri uporabe združeni. Združeni primeri uporabe so Administracija upo-

rabnikov, Administracija šifrantov lokacij root, Določitev pravic za dostop do lokacij root in Vpogled v dnevnik prenosov.

Iz diagrama primerov uporabe je razvidno, da imamo dva tipa akterjev, to sta Uporabnik in Administrator. Uporabnik je stranka podjetja, ki lahko preko spleta pregleduje dnevnik prenosov datotek (primer uporabe: Vpogled v dnevnik prenosov). Administrator je oseba, ki ima pravico do vseh funkcij v sistemu.

Uporabniške zahteve za spletno stran adiminstracije strežnika so bile naslednje:

- administracija uporabnikov,
- administracija šifrantov tako imenovanih lokacij root (pot, ki je dosegljiva s strežnika),
- določitev pravic za dostop do lokacij root na uporabnika in
- dnevnik prenosov.

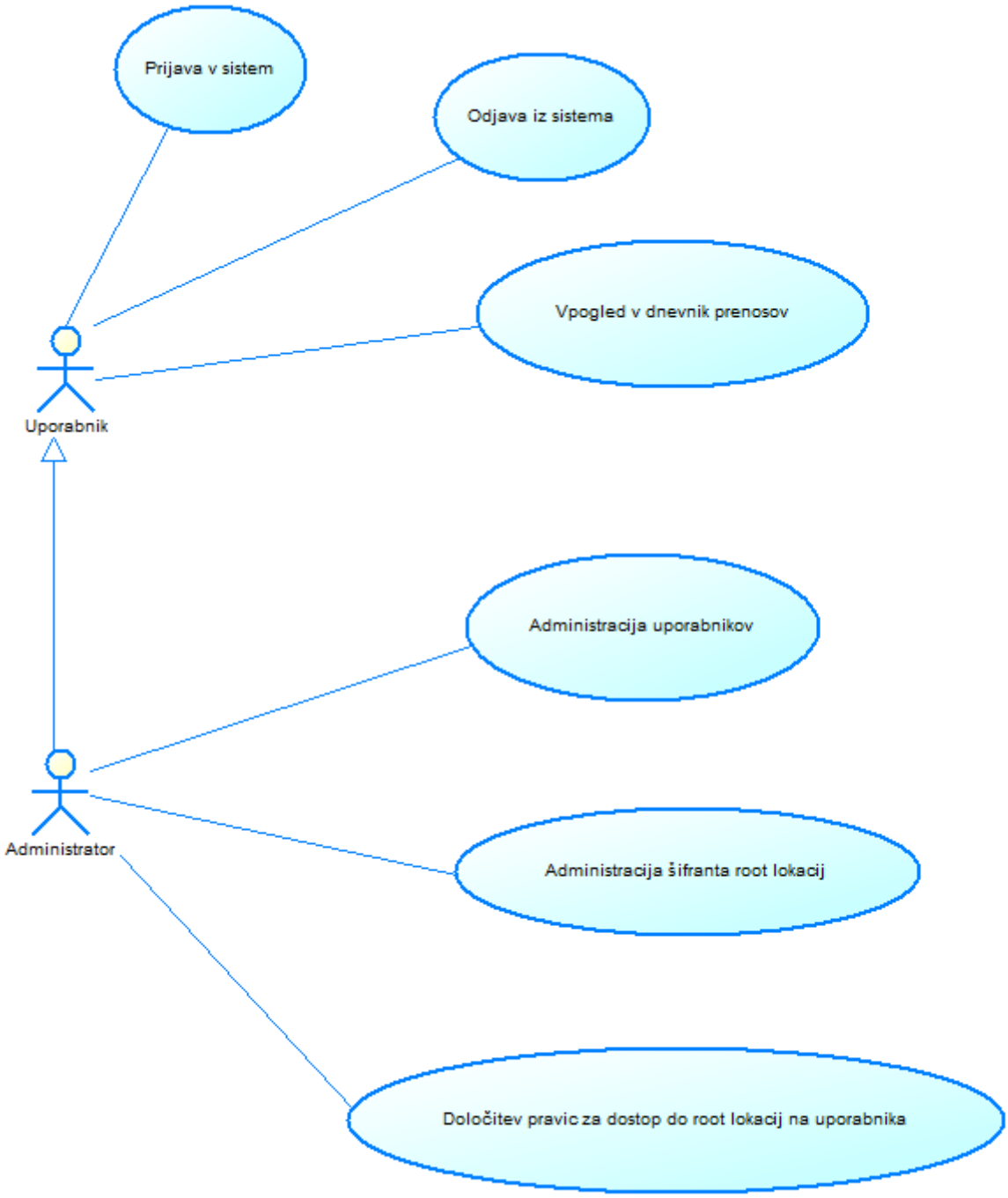
Uporabniške zahteve za administracijo uporabnikov so razvidne s slike 3.2. Administrator ima tako pravice dodajanja in brisanja uporabnikov. Prav tako dodeljuje pravice in pripadajoče lokacije na strežniku.

Uporabniške zahteve za administracijo šifrantov so razvidne s slike 3.3. Administrator lahko briše, dodaja in/ali ureja šifrante.

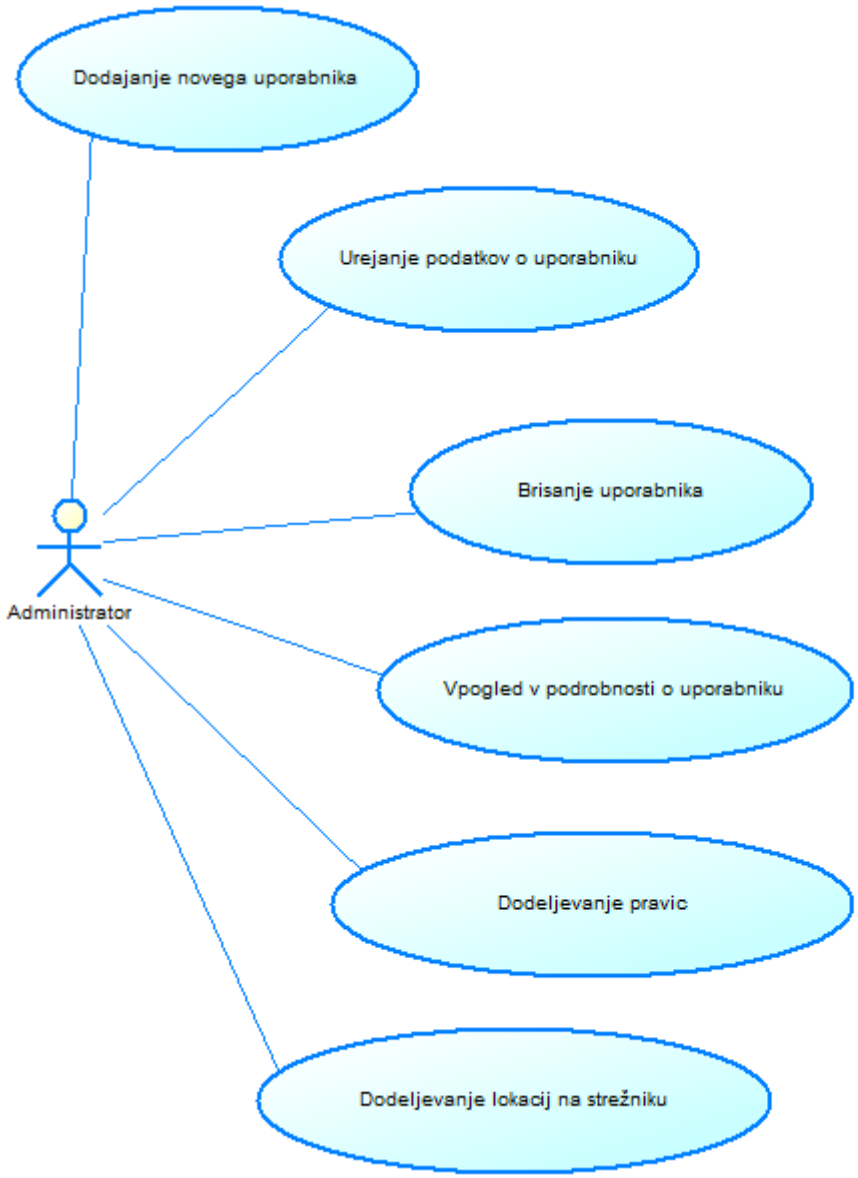
V okviru dnevnika prenosov je bila podana zahteva, da se evidentirajo vsi uspešni in neuspešni prenosi ter možnost pregleda dnevnika in izvoza podatkov (slika 3.4).

3.1.2 Spletna storitev

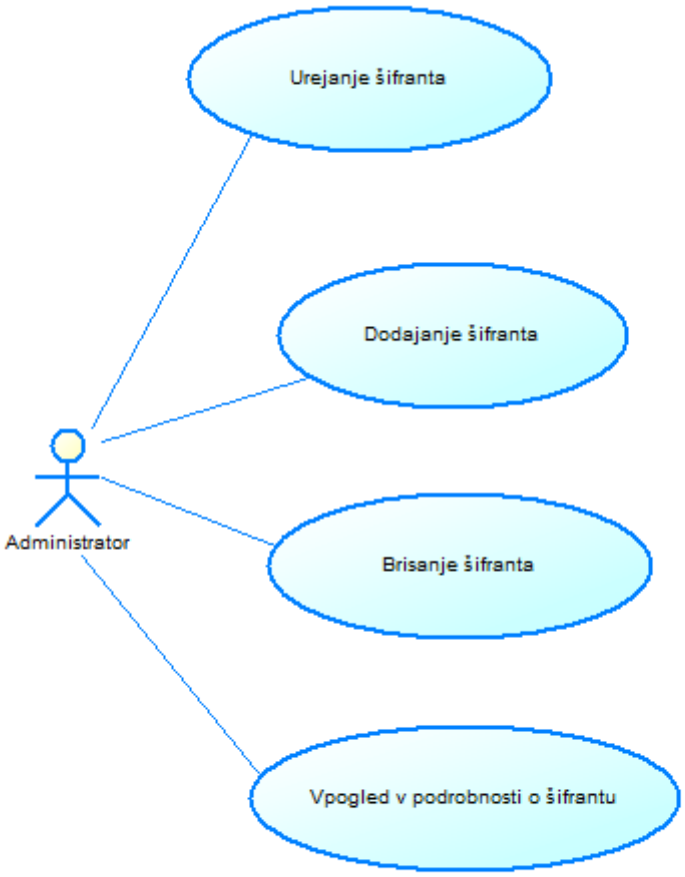
Glavne funkcionalnosti, ki jih mora podpirati spletna storitev za izmenjavo datotek, so pridobitev seznama datotek in direktorijske strukture na poljubni lokaciji root, možnost prenosa celotne direktorijske strukture z ene na drugo lokacijo in možnost nadaljevanja prekinjenih datotek. Prenos z ene na drugo



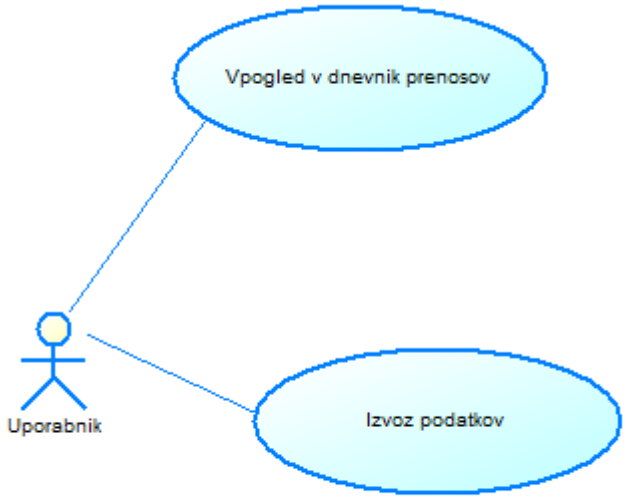
Slika 3.1: Skupni model primerov uporabe



Slika 3.2: Razčlenjeni diagram primerov uporabe Administracije uporabnikov



Slika 3.3: Razčlenjeni diagram primerov uporabe Administracije šifrantov



Slika 3.4: Razčlenjeni diagram primerov uporabe Dnevnika prenosov

lokacijo se izvaja na način prenesi in izbriši iz izvora. Za razvoj spletne storitve je bil izbran generični HTTP-upravljavec. Glavna metoda upravljavca je ProcesRequest, ki sprejme objekt tipa HTTPContext, ki je skupek vseh HTTP-informacij o prejeti zahtevi. Treba je bilo definirati nova polja v glavi HTTP. V tabeli 3.1 so opisana vsa na novo definirana polja.

Ime polja	Tip
`Action`	Tip akcije, ki naj jo izvede spletna storitev
`Path`	Pot lokacije, na katero se akcija navezuje
`StorageRootID`	Identifikacijska številka lokacije na strežniku, ki je zapisana v bazi
`SourceFile`	Izvorna datoteka
`DestinationFile`	Ponorna datoteka
`Mask`	Iskalna maska
`Recursive`	Rekurzivno iskanje po direktoriju
`Checksum`	Računanje kontrolne vsote

Tabela 3.1: Na novo definirana polja v glavi HTTP

S pomočjo na novo definiranih polj lahko sporočimo upravljavcu, kakšen tip akcije je treba izvesti. Spletna storitev sprejema zahteve (akcije) odjemalca in jih izvaja. Akcije so tipa:

- delete - brisanje izbrane datoteke s strežnika,
- upload - prenos datoteke od odjemalca na strežnik,
- download - prenos datoteke s strežnika na odjemalca,
- move - preimenovanje datotek po končanem prenosu,
- dir - seznam vseh datotek na izbrani lokaciji,
- checksum - kontrolna vsota dokumenta,
- begintransaction - seznam vseh lokacij uporabnika na strežniku in
- getfolders - prikaz datotečne strukture na strežniku.

3.2 Odjemalec

3.2.1 Servisni modul

Servisni modul mora omogočati več različnih konfiguracij za prenos datotek. Vsaka konfiguracija je svoj posel. Omogočati mora t.i. scheduling, kar pomeni, da se servisni modul lahko izvaja na intervale ali na določene ure v dnevu, tednu ali mesecu. Omogočati mora filter na datoteke, ki jih želimo prenašati, in prenos celotne direktorijske strukture. Prav tako je za potrebe obveščanja o delovanju servisnega modula treba realizirati obveščanje preko elektronske pošte o uspešnih in neuspešnih prenosih. Obveščanje deluje na nivoju vseh poslov oziroma na nivoju posameznega posla.

3.2.2 Uporabniški vmesnik za konfiguracijo servisnega modula

Z uporabniškim vmesnikom lahko hitro in enostavno skonfiguriramo servisni modul. Vse nastavitve servisnega modula se zapišejo v datoteko xml. Zajema vse nastavitve, ki so potrebne za pravilno delovanje servisnega modula. Uporabniški vmesnik mora vsebovati naslednje štiri nastavitve:

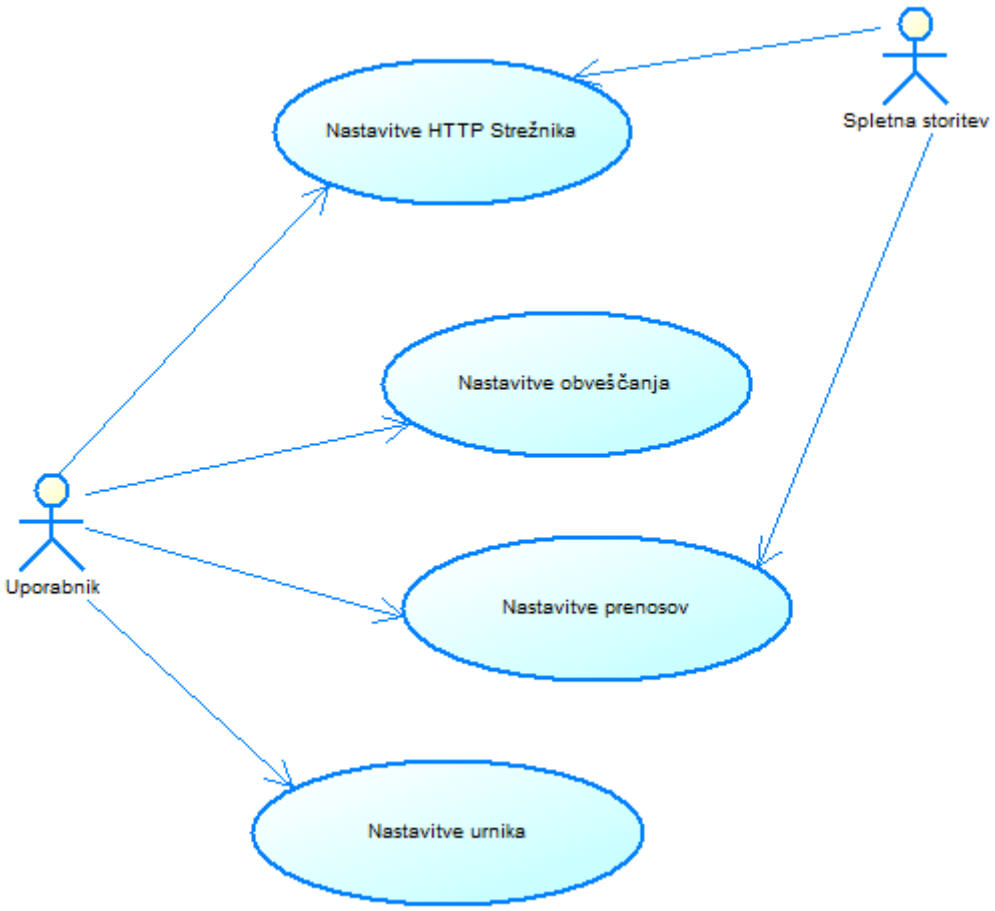
- nastavitve HTTP-strežnika,
- nastavitve obveščanja,
- nastavitve tipov prenosov in
- nastavitve urnika.

Na sliki 3.5 je predstavljen diagram primerov uporabe za uporabniški vmesnik. Pri tem so vsi primeri uporabe sestavljeni. Zaradi boljše preglednosti so združeni. Iz diagrama primerov uporabe je razvidno, da sodelujeta dva akterja: Uporabnik in Spletna storitev. Uporabnik je začetnik primera uporabe (uporablja določeno funkcionalnost sistema). Akter Spletna storitev pa le sodeluje pri določenem primeru uporabe. V mojem primeru sodeluje pri dveh primerih uporabe: Nastavitve HTTP-strežnika in Nastavitve prenosov. Pri nastavitvah HTTP-strežnika se kliče spletna storitev, ki preveri, ali so vsi podatki za dostop do spletne storitve točni (avtentikacija uporabnika). Pri nastavitvah prenosov se spletna storitev kliče v primeru izpisa direktorijske strukture na strežniku ter pri pridobitvi seznama vseh lokacij root določenega uporabnika na strežniku.

Nastavitve HTTP-strežnika vsebujejo nastavitve URL-naslova, nastavitve avtentikacije in strežnika proxy.

Nastavitve obveščanja vsebujejo dodajanje, brisanje in posodabljanje prejemnikov.

Nastavitve urnika omogočajo dodajanje, brisanje, preimenovanje in posodabljanje urnikov.

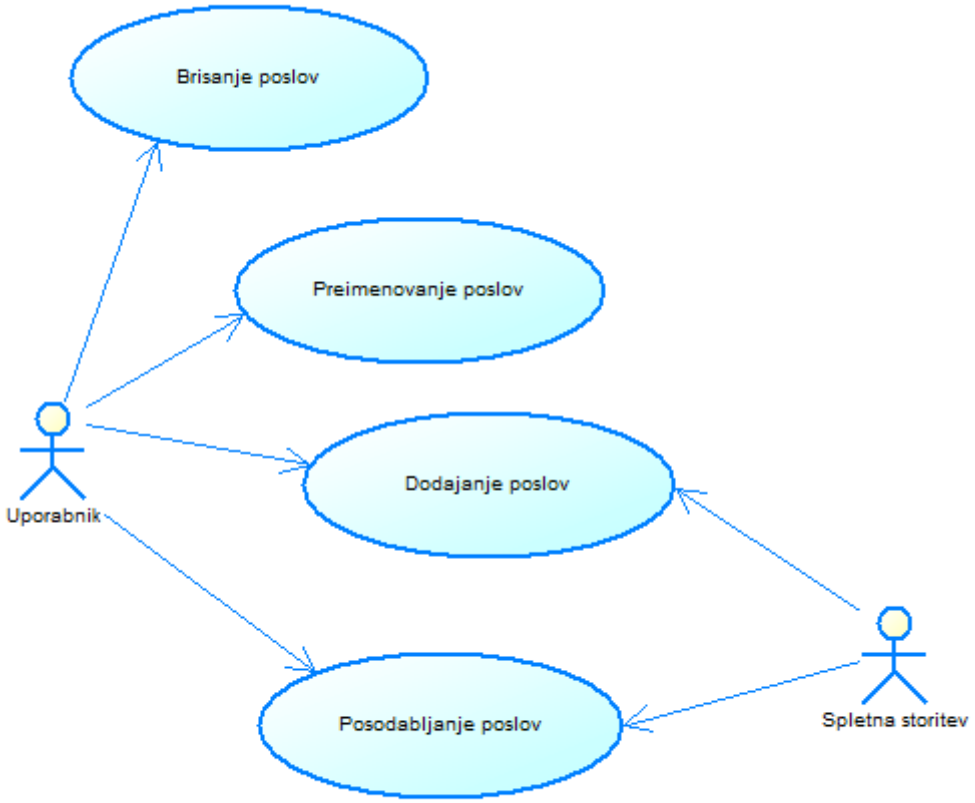


Slika 3.5: Sestavljeni diagram primerov uporabe za nastavitve uporabniškega vmesnika

Iz diagrama primerov uporabe 3.6 lahko razberemo, katere zahteve so bile podane: brisanje, dodajanje, posodabljanje in preimenovanje poslov. Posel je v tem primeru en tip prenosa. Pri dodajanju poslov so bile podane naslednje zahteve:

- izbira lokacije root,
- izbira tipa smeri prenosa,
- izbira direktorija,
- izbira maske,
- izbira urnika in
- nastavitve obveščanja.

Izbira lokacije root je potrebna, saj ima lahko uporabnik več dodeljenih lokacij na strežniku. Pri izbiri lokacije sodelujeta dva akterja. Spletna storitev omogoča prikaz vseh lokacij na strežniku. Uporabnik je začetnik primera uporabe. Imamo dva tipa smeri prenosa: upload in download. Upload je prenos, pri katerem prenašamo datoteko od izvorne lokacije uporabnika do ponorne lokacije na strežniku. Download je prenos, pri katerem prenašamo datoteko od izvorne lokacije na strežniku do ponorne lokacije odjemalca. Pri izbiri direktorija sodelujeta dva akterja: Uporabnik in Spletna storitev. Uporabnik je začetnik primera uporabe, medtem ko spletna storitev sodeluje pri primeru uporabe. Spletna storitev omogoča izbiro direktorija na strežniški strani. Vsak posel ima tudi urnik. Z izbiro urnika določimo poslu termin izvajanja. Po direktoriju, iz katerega bomo na strežnik ali z njega izvajali prenos, lahko iščemo rekurzivno (iščemo tudi po poddirektorijih) ali nerekurzivno (iščemo le po trenutnem direktoriju). Potrebna je tudi izbira maske, preko katere določimo tip datotek, ki jih bomo prenašali (npr. za prenos datotek zip je maska enaka *.zip). Na koncu je treba določiti še nastavitve obveščanja, kot so naslovi prejemnikov in tip obveščanja. Tipi obveščanja so štirje, in sicer:



Slika 3.6: Razčlenjeni diagram primera uporabe Nastavitve prenosa

- le ob uspešnih prenosih,
- le ob neuspešnih prenosih,
- na nivoju vseh poslov in
- na nivoju posameznega posla.

Vsak prejemnik je lahko omogočen ali onemogočen. Vsak posel je prav tako lahko omogočen ali onemogočen.

Vsak posel lahko brišemo, posodabljamo ali preimenujemo.

Poglavje 4

Analiza in načrtovanje

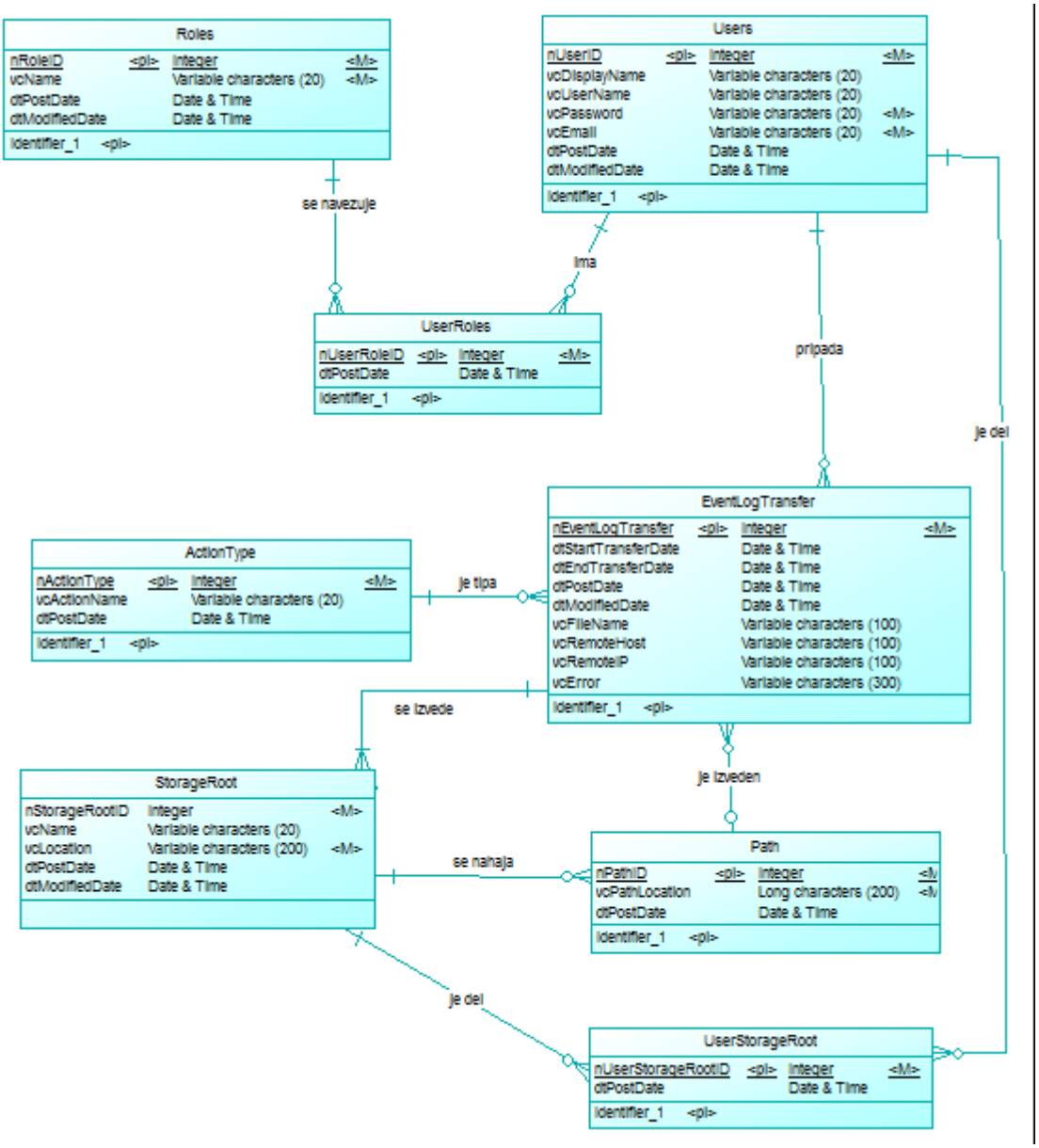
V analizi je predstavljen konceptualni in logični model. Nato sledijo še razredni diagram in diagrami zaporedja. Za predstavitev celotnega sistema je na koncu definiran še diagram postavitve, ki prikazuje celotno konfiguracijo sistema.

4.1 Konceptualni model

V tem poglavju je predstavljen konceptualni model. Konceptualni model vsebuje entitete, ki jim določimo primarni ključ in atribute. Prav tako med entitetami določimo razmerja. Konceptualni model sestoji iz osmih entitet. Za numerične tipe atributov se pred imenom atributa zapiše črka n kot oznaka za numeric (npr. nKoličina). Pred tipom DateTime se zapiše dt (npr dtČasPrenosa). Pred tipom varchar pa se uporabljata črki vc (npr. vcOpisNapake).

Na sliki 4.1 so naslednje entitete:

- Roles,
- Users,
- UserRoles,
- EventLogTransfer,



Slika 4.1: Konceptualni model

- ActionType,
- StorageRoot,
- Path in
- UserStorageRoot.

4.2 Logični model

Na sliki 4.2 je predstavljen logični model, ki je bil zgeneriran iz konceptualnega modela na sliki 4.1. Logični podatkovni model predstavlja podatkovno strukturo v taki obliki, kot bo implementirana v podatkovni bazi. Kreiramo ga za določen SUPB (Sistem za upravljanje s podatkovno bazo). V mojem primeru je bil uporabljen SUPB MS SQL.

4.2.1 Entiteta Users

V entiteti Users so shranjene informacije o uporabnikih, ki imajo dostop do spletnega vmesnika za administracijo strežnika in lahko uporabljajo spletno storitev za izmenjavo datotek. O njih hranimo osnovne informacije. V tabeli 4.1 so predstavljeni vsi atributi entitete.

Ime atributa	Tip	Opis polja
`nUserID`	int	Identifikator uporabnika
`vcDisplayName`	nvarchar(20)	Prikazno ime uporabnika
`vcUserName`	nvarchar(20)	Uporabniško ime uporabnika
`vcPassword`	nvarchar(20)	Geslo uporabnika
`vcEmail`	nvarchar(20)	Elektronski naslov uporabnika
`dtPostDate`	DateTime	Datum zapisa v podatkovno bazo
`dtModifiedDate`	DateTime	Datum spremembe zapisa v podatkovno bazo

Tabela 4.1: Opis atributov entitete Users

4.2.2 Entiteta StorageRoot

V entiteti StorageRoot so shranjene lokacije na strežniku, do katerih uporabnik lahko dostopa. To pomeni, da lahko s te lokacije ali nanjo prenaša dokumente. Vsak uporabnik ima lahko eno ali več lokacij root. Katere lokacije pripadajo kateremu uporabniku, pa določa entiteta UserStorageRoot. V tabeli 4.2 so predstavljeni vsi atributi entitete.

Ime atributa	Tip	Opis polja
`nStorageRootID`	int	Identifikator lokacije root
`vcName`	nvarchar(20)	Ime lokacije
`vcLocation`	nvarchar(20)	Pot lokacije
`dtPostDate`	DateTime	Datum zapisa v podatkovno bazo
`dtModifiedDate`	DateTime	Datum spremembe zapisa v podatkovno bazo

Tabela 4.2: Opis atributov entitete StorageRoot

4.2.3 Entiteta UserStorageRoot

V tabeli 4.3 so predstavljeni vsi atributi entitete UserStorageRoot, ki je vmesna tabela med uporabniki in lokacijami storage root.

Ime atributa	Tip	Opis polja
`nUSerStorageRootID`	int	Identifikator lokacije root uporabnika
`nUserID`	nvarchar(20)	Identifikator uporabnika
`nStorageRootID`	nvarchar(20)	Identifikator lokacije root
`dtPostDate`	DateTime	Datum zapisa v podatkovno bazo

Tabela 4.3: Opis atributov entitete UserStorageRoot

4.2.4 Entiteta EventLogTransfer

Entiteta EventLogTransfer ali Dnevnik prenosov shranjuje informacije o uspešnih in neuspešnih prenosih. Prav tako beleži vsako akcijo, ki se izvede nad določeno lokacijo root. Za vsak prenos se beleži tudi začetni in končni čas

prenosa. Ob neuspešnem prenosu se zabeleži tudi kratek opis napake. Vsi atributi entitete so opisani v tabeli 4.4.

Ime atributa	Tip	Opis polja
nEventLogTransferID	int	Identifikator prenosa
vcRemoteHost	nvarchar(100)	Ime gostitelja
vcRemoteIP	nvarchar(100)	IP-naslov gostitelja
nUserID	int	Identifikator uporabnika
nStorageRootID	int	Identifikator lokacije root
nPathID	int	Identifikator poti
vcFileName	nvarchar(100)	Ime datoteke
nActionType	int	Identifikator akcije
dtStartTransferDate	DateTime	Datum in čas začetka prenosa
dtEndTransferDate	DateTime	Datum in čas konca prenosa
dtPostDate	DateTime	Datum zapisa v podatkovno bazo
dtModifiedDate	DateTime	Datum spremembe zapisa v podatkovno bazo
vcError	nvarchar(300)	Opis napake pri prenosu

Tabela 4.4: Opis atributov entitete EventLogTransfer

4.2.5 Entiteta ActionType

Vsak prenos ima definiran tip akcije. Atributi so opisani v tabeli 4.5. Tipi akcije, ki so definirani, so: upload, download, move, delete, dir.

Ime atributa	Tip	Opis polja
nActionTypeID	int	Idenifikator akcije
vcActionName	nvarchar(20)	Ime akcije
dtPostDate	DateTime	Datum zapisa v podatkovno bazo

Tabela 4.5: Opis atributov entitete ActionType

4.2.6 Entiteta Path

Prenos je definiran tudi s potjo, kjer se posamezna akcija izvaja. Opis atributov je opisan v tabeli 4.6. Če se akcija izvede nad lokacijo root, potem prenos nima definirane poti. Pot je lokacija na strežniku od lokacije storage root naprej.

Ime atributa	Tip	Opis polja
`nPathID`	int	Idenifikator akcije
`vcPathLocation`	nvarchar(20)	Ime akcije
`nStorageRootID`	integer	Indenifikator lokacije root
`dtPostDate`	DateTime	Datum zapisa v podatkovno bazo

Tabela 4.6: Opis atributov entitete Path

4.2.7 Entiteta Roles

V tej entiteti so shranjene vse pravice, ki jih lahko določimo uporabniku. V tabeli 4.7 so opisani vsi atributi. Vsak uporabnik ima lahko nič ali več pravic. Ob kreiranju novega uporabnika se mu določi le pravica vpogleda v dnevnik prenosov. Dodatne pravice mu nastavlja administrator.

Ime atributa	Tip	Opis polja
`nRoleID`	int	Idenifikator pravice
`vcName`	nvarchar(20)	Ime pravice
`dtModifiedDate`	DateTime	Datum spremembe zapisa v podatkovno bazo
`dtPostDate`	DateTime	Datum zapisa v podatkovno bazo

Tabela 4.7: Opis atributov entitete Roles

4.2.8 Entiteta UserRoles

Ta entiteta shranjuje vse pravice uporabnika. V tabeli 4.8 so opisani atributi. UserRoles je vmesna tabela med entitetama Roles in Users.

Ime atributa	Tip	Opis polja
`nUserRolesID`	int	Idenifikator pravice
`nUserID`	nvarchar(20)	Identifikator uporabnika
`nRoleID`	DateTime	Identifikator pravice
`dtPostDate`	DateTime	Datum zapisa v podatkovno bazo

Tabela 4.8: Opis atributov entitete Roles

4.3 Razredni diagram - spletna stran za administracijo strežnika

Predstavljen je razredni diagram za spletno stran administracije strežnika. Pri prikazu razredov so uporabljeni stereotipi «control», «entity» in «boundary». Kontrolni razred «control» koordinira dogajanje znotraj primera uporabe. Kontrolni razredi učinkovito ločijo mejne in poslovne razrede ter s tem naredijo sistem odporen na spremembe. Tipičen primer kontrolnih razredov je npr. razred za obvladovanje napak. Poslovni razred «entity» je od okolja neodvisen in hrani ter upravlja podatke. Poslovni razred prav tako ponavadi ni vezan le na en primer uporabe. Mejni razred «boundary» je posrednik med okoljem in sistemom ter je odvisen od sprememb v sistemu. Primer mejnega razreda so razredi uporabniškega vmesnika.[6]

Na sliki 4.3 je predstavljen razredni diagram za primer uporabe Administracije uporabnikov. S slike je razvidno, da ima razredni diagram sedem mejnih razredov, en kontrolni razred in štiri poslovne razrede. Kontrolni razred UserController je skupen za celoten diagram primerov uporabe in vsebuje vse potrebne operacije za implementacijo primera uporabe. Operacije, ki jih vsebuje, so:

- Index (operacija vrne seznam vseh uporabnikov oziroma prikaže uporabnika glede na iskalni niz),
- Details (operacija vrne podrobne informacije o uporabniku, ki smo ga izbrali s seznama vseh uporabnikov),

- Create (operacija ustvari novega uporabnika),
- Edit (operacija vrne izbranega uporabnika),
- EditConfirmed (operacija spremememeni posodobljene informacije o uporabniku),
- Delete (operacija vrne podrobnosti o izbranem uporabniku),
- DeleteConfirmed (operacija izbriše uporabnika),
- Permissions (operacija vrne seznam vseh dodeljenih in nedodeljenih pravic uporabnika),
- PermissionsConfirmed (operacija dodeli nove pravice uporabniku),
- Storage (operacija vrne seznam dodeljenih in nedodeljenih lokacij na strežniku) ter
- StorageConfirmed (operacija uporabniku dodeli nove lokacije na strežniku).

4.4 Diagram zaporedja

4.4.1 Diagrami zaporedja za administracijo strežnika

Na sliki 4.4 je podan diagram zaporedja Vnos novega uporabnika. V tem diagramu zaporedja sodelujejo zaslonska maska UserCreate, kontrolni razred UserController in entitetni razred Users. Diagram prikazuje, kako poteka vnos novega uporabnika. Najprej uporabnik vpiše vse potrebne informacije. Ob klicu Dodaj se kliče kontrolni razred, ki preveri vse vpisane podatke in izvede vpis novega uporabnika v podatkovno bazo.

Na sliki 4.5 je podan diagram zaporedja Dodeljevanje pravic uporabniku. V tem diagramu zaporedja sodelujejo zaslonska maska UserPermissions, kontrolni razred UserController ter dva entitetna razreda Permissions in

UserPermissions. Diagram prikazuje potek shranjevanja pravic uporabnika. Uporabniku se najprej prikaže seznam vseh dodeljenih in nedodeljenih pravic. Uporabnik nato izbere želene pravice. Na koncu se dodeljene pravice shranijo v podatkovno bazo.

4.4.2 Diagrami zaporedja spletnih storitev

Na sliki 4.6 je podan diagram zaporedja za prenos datoteke na strežnik. V tem diagramu sodelujeta dve komponenti: klient in spletni strežnik. Spletni strežnik je sestavljen iz spletne storitve in podatkovne baze. Klient pa iz servisnega modula, ki je za boljšo predstavitev razdeljen na dva dela: Servisni modul in upload. Funkcija upload je zajeta v servisnem modulu, vendar je definirana kot samostojna komponenta za boljšo preglednost diagrama. Izpisane so akcije, do katerih pride ob prenosu datoteke na strežnik. Prva akcija je Login, ki omogoča pridobitev avtentikacijskega piškotka za potrebe avtentikacije klienta na spletnem strežniku. Diagram prikazuje akcije za prenos ene datoteke. V resnici se od akcije IzvrsiUpload naprej vse izvaja v zanki toliko časa, dokler niso prenesene vse datoteke s podane lokacije posla. Vsaka zahteva, ki je poslana na spletni strežnik, se zapiše v podatkovno bazo. Prenos datoteke na strežnik se končna z brisanjem datoteke na klientu.

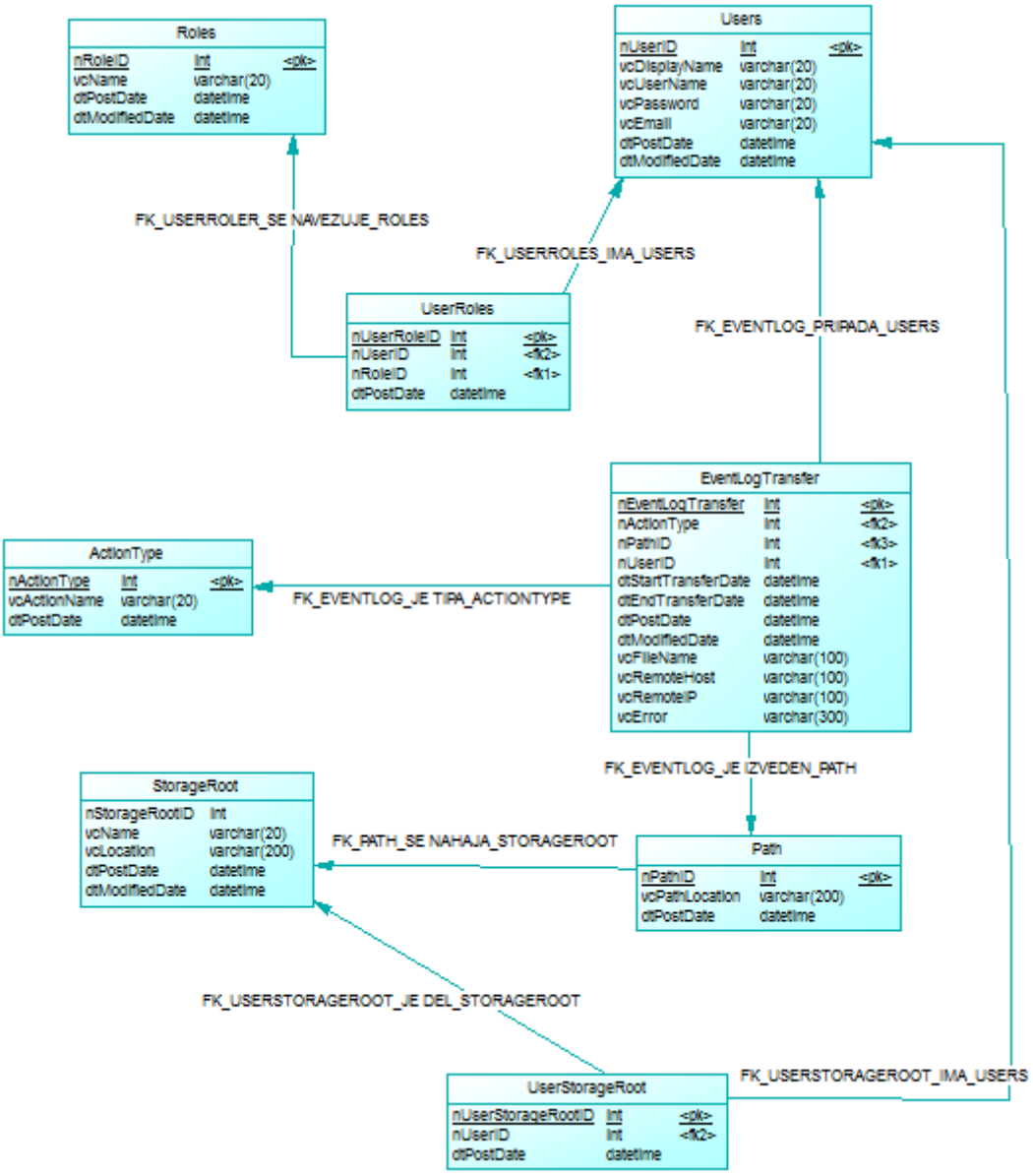
4.5 Diagram postavitve

Na sliki 4.7 je prikazan diagram postavitve sistema. Diagram postavitve sistema prikaže konfiguracijo sistema in postavitve komponent, ki se izvajajo na vozliščih. Vozlišče je fizični element, ki predstavlja računalniški vir. Komponenta je fizični del sistema, ki vsebuje realizacijo množice vmesnikov. Za delovanje sistema morajo biti prisotna naslednja vozlišča:

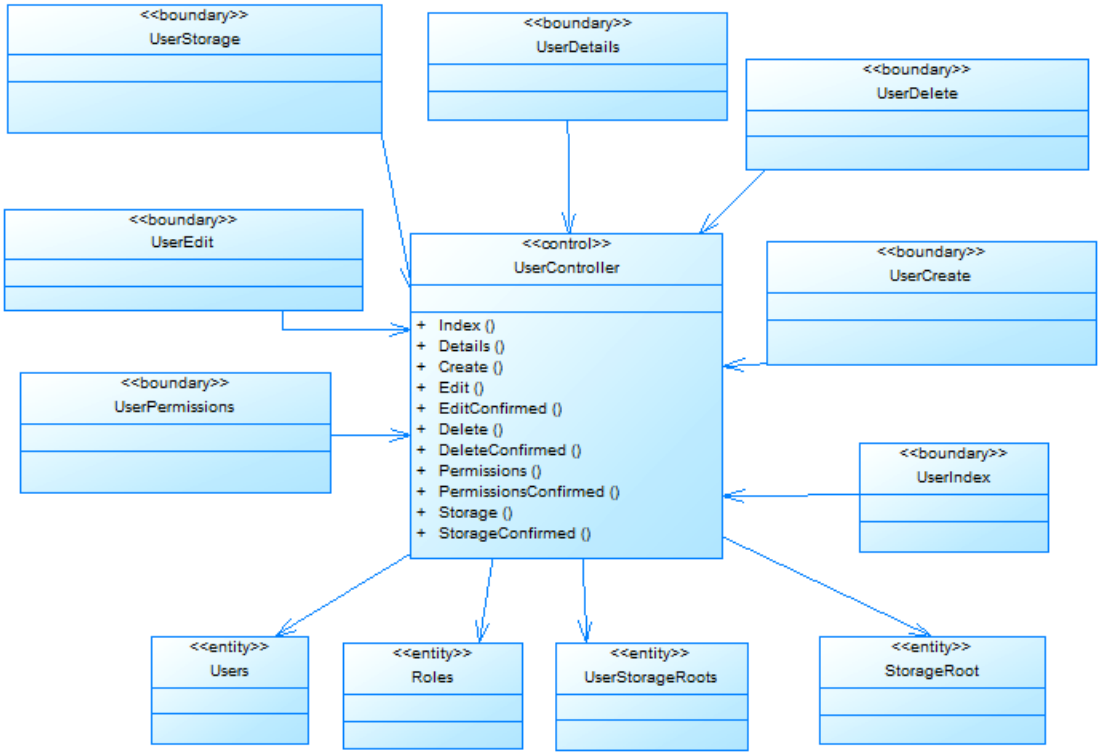
- uporabnikov PC,
- podatkovni strežnik,
- spletni strežnik in

- spletni uporabnik.

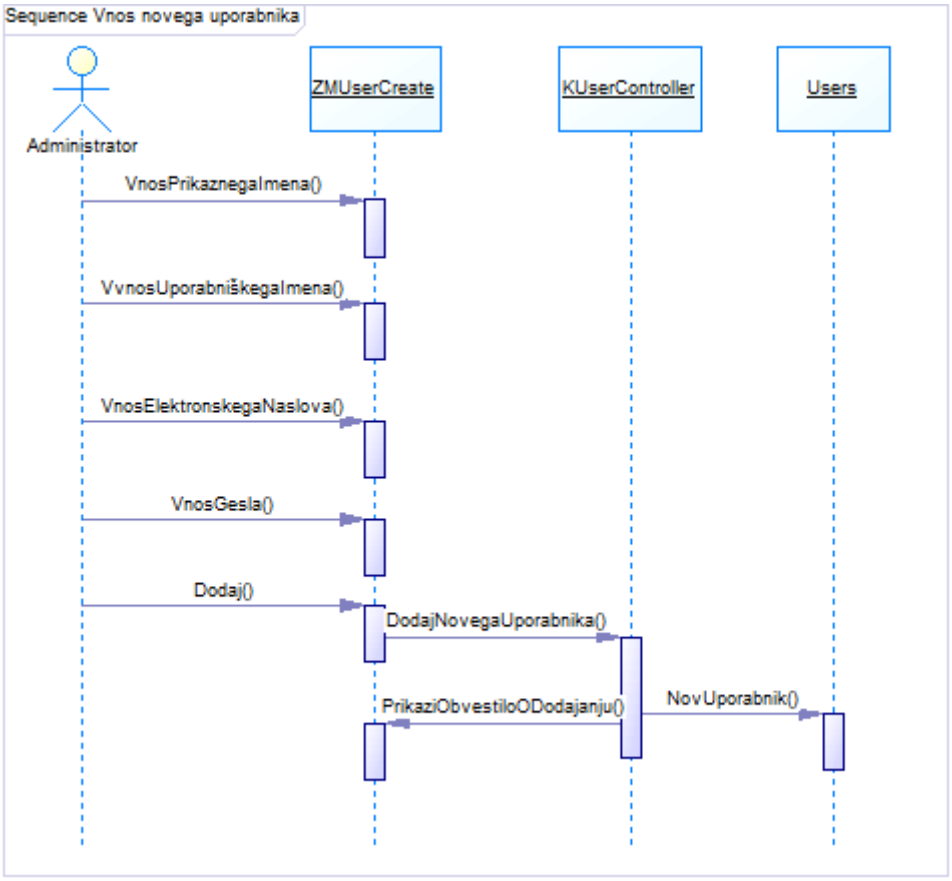
Preko uporabnikovega osebnega računalnika, na katerem teče servisni modul, se datoteke prenašajo s strežnika na odjemalčev PC in obratno. Spletni strežnik je povezan s podatkovnim strežnikom. Na spletnem strežniku sta dve komponenti: spletna stran in spletna storitev za izmenjavo datotek. Ti dve komponenti potrebujeta za nemoteno delovanje dostop do podatkovnega strežnika. Preko spletnega brskalnika lahko dostopamo do spletne strani za administracijo strežnika.



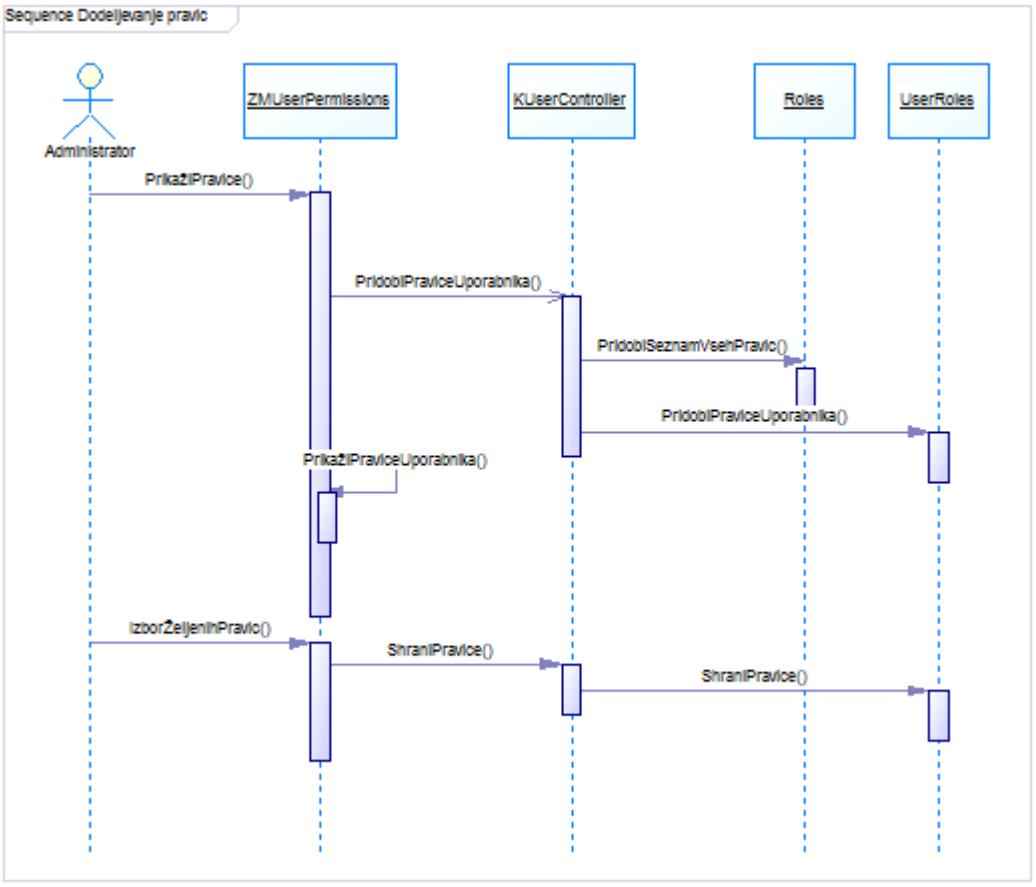
Slika 4.2: Logični model



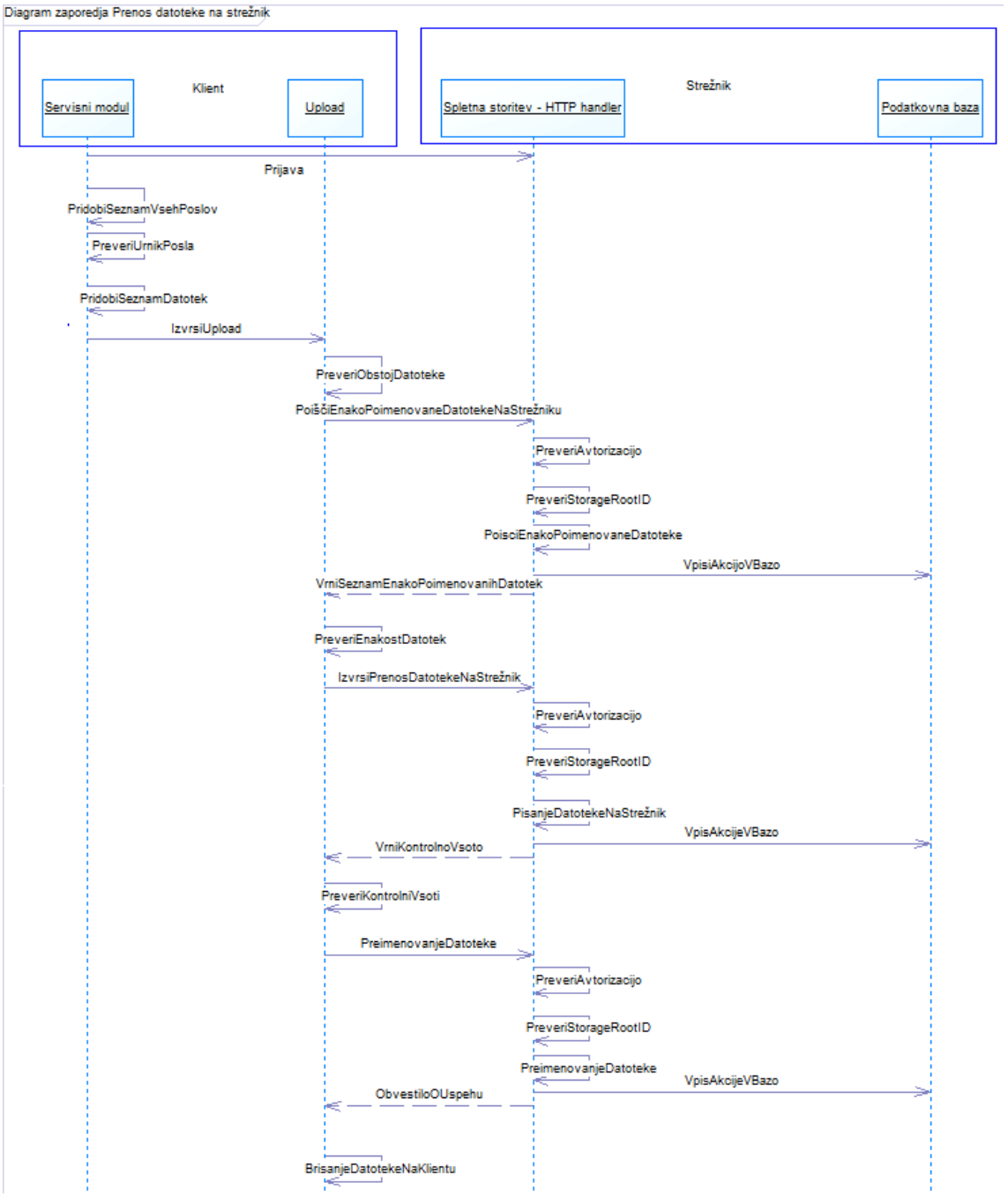
Slika 4.3: Razredni diagram za primer uporabe Administracije uporabnikov



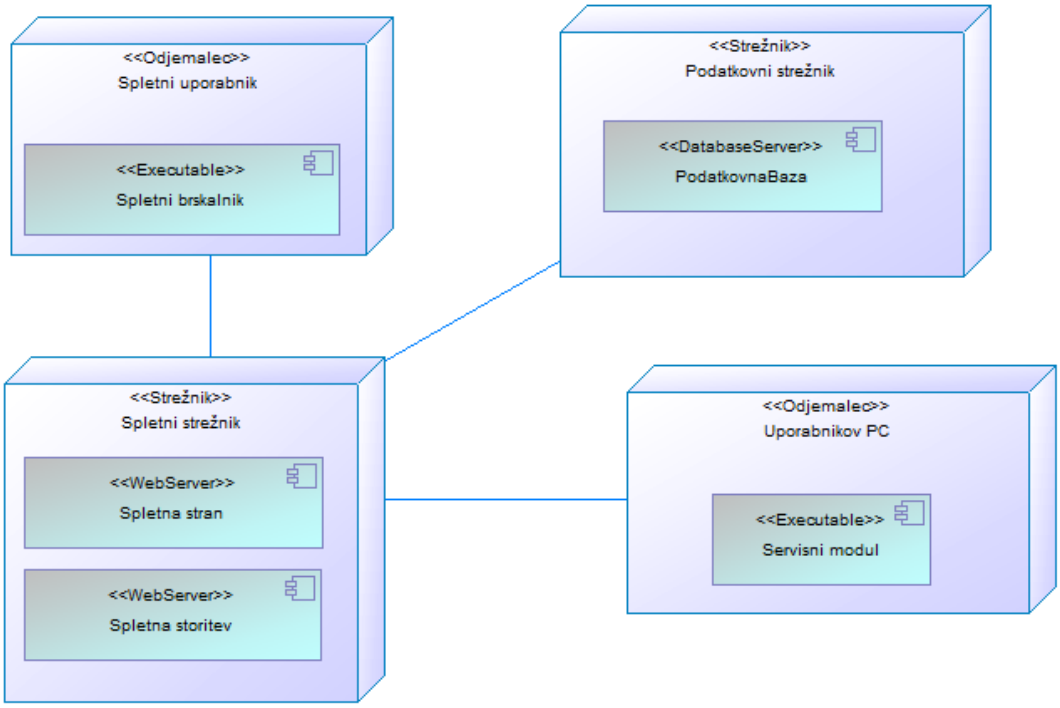
Slika 4.4: Diagram zaporedja za primer uporabe Administracije uporabnikov



Slika 4.5: Diagram zaporedja za primer uporabe Dodeljevanja pravic uporabnika



Slika 4.6: Diagram zaporedja za Prenos datoteke na strežnik



Slika 4.7: Diagram postavitve

Poglavje 5

Implementacija

Za implementacijo produkta je bil uporabljen programski jezik C#.

Sama implementacije je potekala v treh korakih, ki so predstavljeni v naslednjih podpoglavjih. Spletna stran in spletna storitev, ki sta na IIS-strežniku, uporabljata protokol HTTPS, ki omogoča varno povezavo.

5.1 Implementacija spletne strani za administracijo strežnika

Prvi korak je bila implementacija spletne strani za administracijo strežnika. Ker je spletna stran narejena v ogrodju MVC, je bila sama implementacija zelo enostavna. V nadaljevanju je predstavljen način za implementacijo po meri narejenih razredov za potrebe avtorizacije in prijavljanja v spletno stran.

Za potrebe avtorizacije je bilo treba ustvariti dva ponudnika: Custom Membership in Custom Role. S ponudnikom Custom Memebership je mogoče avtorizirati uporabnike in zaščititi dele ali celotne strani. Na sliki 5.1 je prikazana implementacija validacije prijave. Validacija prijave se izvede s povezavo na podatkovno bazo, kjer se pridobi geslo uporabnika. Geslo pridobljeno iz podatkovne baze ima t.i. zgoščeno vrednost, ki jo potem primerjamo s zgoščeno vrednostjo gesla, ki ga je uporabnik vpisal. Če se gesli ujemata se je uporabnik uspešno prijavil v spletno aplikacijo.

```
public override bool ValidateUser(string username, string password)
{
    if (db.Users.Where(x => x.vcUsername == username) != null)
    {
        Users user = db.Users.Where(x => x.vcUsername == username).FirstOrDefault();
        if (user != null)
        {
            if (user.vcPassword == HashHelper.ComputeHashSHA1(password))
                return true;
        }
    }
    return false;
}
```

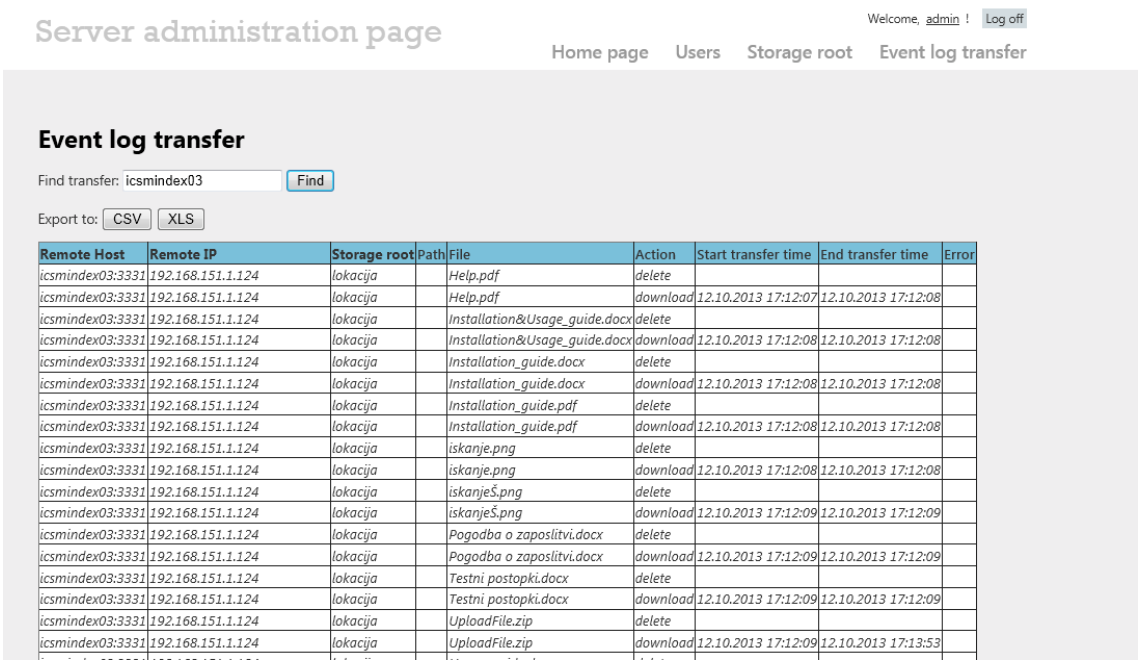
Slika 5.1: Izsek kode za validacijo prijave

```
public override string[] GetRolesForUser(string username)
{

    List<string> result = new List<string>();

    Users usr = db.Users.Single(u => u.vcUsername.Equals(username, StringComparison.InvariantCultureIgnoreCase));
    List<UserRoles> usrroles = db.UserRoles.Where(x => x.nUserID == usr.nUserID).ToList();
    foreach(UserRoles item in usrroles) {
        result.Add(item.Roles.vcName);
    }
    return result.ToArray();
}
```

Slika 5.2: Metoda, ki vrne seznam uporabnikov pravic



Remote Host	Remote IP	Storage root	Path	File	Action	Start transfer time	End transfer time	Error
icsmindex03:3331	192.168.151.1.124	lokacija		Help.pdf	delete			
icsmindex03:3331	192.168.151.1.124	lokacija		Help.pdf	download	12.10.2013 17:12:07	12.10.2013 17:12:08	
icsmindex03:3331	192.168.151.1.124	lokacija		Installation&Usage_guide.docx	delete			
icsmindex03:3331	192.168.151.1.124	lokacija		Installation&Usage_guide.docx	download	12.10.2013 17:12:08	12.10.2013 17:12:08	
icsmindex03:3331	192.168.151.1.124	lokacija		Installation_guide.docx	delete			
icsmindex03:3331	192.168.151.1.124	lokacija		Installation_guide.docx	download	12.10.2013 17:12:08	12.10.2013 17:12:08	
icsmindex03:3331	192.168.151.1.124	lokacija		Installation_guide.pdf	delete			
icsmindex03:3331	192.168.151.1.124	lokacija		Installation_guide.pdf	download	12.10.2013 17:12:08	12.10.2013 17:12:08	
icsmindex03:3331	192.168.151.1.124	lokacija		iskanje.png	delete			
icsmindex03:3331	192.168.151.1.124	lokacija		iskanje.png	download	12.10.2013 17:12:08	12.10.2013 17:12:08	
icsmindex03:3331	192.168.151.1.124	lokacija		iskanjeŠ.png	delete			
icsmindex03:3331	192.168.151.1.124	lokacija		iskanjeŠ.png	download	12.10.2013 17:12:09	12.10.2013 17:12:09	
icsmindex03:3331	192.168.151.1.124	lokacija		Pogodba o zaposlitvi.docx	delete			
icsmindex03:3331	192.168.151.1.124	lokacija		Pogodba o zaposlitvi.docx	download	12.10.2013 17:12:09	12.10.2013 17:12:09	
icsmindex03:3331	192.168.151.1.124	lokacija		Testni postopki.docx	delete			
icsmindex03:3331	192.168.151.1.124	lokacija		Testni postopki.docx	download	12.10.2013 17:12:09	12.10.2013 17:12:09	
icsmindex03:3331	192.168.151.1.124	lokacija		UploadFile.zip	delete			
icsmindex03:3331	192.168.151.1.124	lokacija		UploadFile.zip	download	12.10.2013 17:12:09	12.10.2013 17:13:53	

Slika 5.3: Vpogled v dnevnik prenosov

Ponudnik Custom Role pa omogoča zaščito določenih delov strani za določene uporabnike. Za minimalno delovanje je bilo treba implementirati metodo GetRolesForUser. Metoda vrne seznam vseh uporabnikovih pravic, ki so zapisane v podatkovni bazi. Na sliki 5.2 je izsek te kode. Z uporabo atributa Authorize lahko zaščitimo celotne strani ali njihove dele in akcije za uporabnike, ki so v specifični vlogi.

Na sliki 5.3 je prikazan vpogled v dnevnik prenosov. Po dnevniku prenosov lahko iščemo oziroma jih sortiramo glede na vse prikaze atribute v tabeli. S klikom na gumb "CSV" se ves seznam prenosov izvozi v t.i. comma-seperated values datoteko. S klikom na gumb "XLS" pa se izvede izvoz vseh podatkov v format primen za Microsoft Excel. Vsak uporabnik vidi le svoje prenose. Administrator pa ima pravico vpogleda v vse prenose.

5.2 Implementacija spletne storitve za izmenjavo datotek

Spletna storitev je implementirana s pomočjo generičnega upravljavca. Glavna metoda ProcessRequest je prikazana na sliki 5.4.

V metodi se naprej preveri avtorizacija uporabnika, ki želi dostopati do spletne storitve. Avtorizacija poteka preko piškotkov. V HTTP-zahtevi se pričakuje piškotek FormsAuthentication, ki nosi zakriptirano vrednost uporabniškega imena. S pomočjo uporabniškega imena dobimo identifikacijsko številko uporabnika, ki služi za nadaljnje delo s HTTP-zahtevo. Vsakič, ko pride zahteva, se izvede proces avtorizacije. Ob neuspešni avtorizaciji uporabnika spletna storitev pošlje odgovor s statusom kode 401. Ob uspešni avtorizaciji sledi pridobitev polja Action, ki nosi vrednost zahtevane akcije na spletni storitvi. Če akcija ni realizirana v spletni storitvi, se pošlje odgovor z opisom napake in statusom kode 501.

V nadaljevanju sta opisani dve najbolj pomembni funkciji spletne storitve Upload in Dowload. Obe funkciji na začetku preverita, ali identifikacijska številka StorageRoot, ki je navedena v zahtevi, pripada uporabniku, ki je zahtevo poslal.

HTTP-polje Range, ki je oblike bytes=x-y, je pomembno za realizacijo zahteve po nadaljevanju prekinjenih prenosov (“resume”). X in y sta v tem primeru vrednosti bajtov, ki naj se prenesejo. Npr. bytes=500-999 pomeni, da se prenaša del entitete od bajte 500 do 999. V tem primeru se bo prineslo 499 bajtov entitete. Glede na vrednost polja Range se izvrši prenos datoteke na strežnik. Na koncu se preveri še kontrolna vsota datotek. Polje s kontrolno vsoto nosi vrednost kontrolne vsote datoteke na klientu. Ta vrednost se nato primerja z vrednostjo kontrolne vsote datoteke na strežniku. Glede na to vrednost se pošlje odgovor klientu. Klient nato skladno z odgovorom izvrši akcije. Če se vrednosti ujemata, se proži akcija preimenovanja - Move. Akcija preimenovanja je potrebna, ker ima zaradi varnosti prenešena datoteka drugačno končnico od originalne datoteke. Na primer, da prenašamo

```
public void ProcessRequest(HttpContext context)
{
    if (CheckAuthorization(context))
    {
        string action = context.Request.Headers["Action"].ToLower();
        //string action = context.Request.QueryString["Action"].ToLower();
        switch (action)
        {
            case "delete":
                Delete(context);
                break;
            case "move":
                Move(context);
                break;
            case "download":
                Download(context);
                break;
            case "upload":
                Upload(context);
                break;
            case "dir":
                Dir(context);
                break;
            case "checksum":
                Checksum(context);
                break;
            case "writeindatabase":
                WriteInDatabaseDownload(context);
                break;
            case "begintransaction":
                BeginTransaction(context);
                break;
            case "getfolders":
                GetFolders(context);
                break;
            default:
                WriteResponse(context, "Action is not supported.", 501);
                break;
        }
    }
    else
        context.Response.StatusCode = 401;
}
```

Slika 5.4: Glavna metoda ProcessRequest generičnega upravljavca

datoteko ImeDatoteke.txt, se le ta na ponorni lokaciji sprva poimenuje kot ImeDatoteke.txt. Ob akciji preimenovanja pa se ustrezno preimenuje. Če se vrednosti ne ujemata, se proži akcija Delete, ki izbriše datoteko s strežnika.

Težave pri funkciji download so se pojavile z metodo TransmitFile. Metoda zapiše datoteko neposredno v odgovor brez shranjevanja v pomnilnik. Sprejme tri atribute: filename (ime datoteke), offset (začetna pozicija v datoteki, s katere se začne prenašati datoteka) in length (število bajtov, ki naj se prenesejo). Funkcija za atributa length in offset sprejme števila v rangu int32. Posledično ta metoda ni primerna za prenos velikih datotek (večjih od 2 GB). Za datoteke, večje od 2 GB, je bilo treba definirati drugo metodo. Nova metoda le bere bajte iz datoteke in jih zapisuje v izhodni tok podatkov HTTP-odgovora.

5.3 Implementacija odjemalca

5.3.1 Servisni modul

Modul deluje kot robot, ki na osnovi nastavitev prenese datoteke od odjemalca na strežnik ali obratno. Modul se izvaja neprestano in glede na interval, ki je določen v nastavitveni datoteki, preverja, ali je opravilo na vrsti za izvedbo ali ne.

Ko se servisni modul zažene, se najprej prebere nastavitvena datoteka, v kateri so zapisana opravila. Opravilo ali TransferType je definirano kot objekt, ki vsebuje vse potrebne informacije za uspešno izveden prenos. Ko se prebere nastavitvena datoteka, se posodobi še urnik. Urnik se posodobi glede na trenutni čas in datum ter tip ponavljanja urnika. Imamo tri tipe ponavljanja urnika: dnevno, mesečno in tedensko ponavljanje.

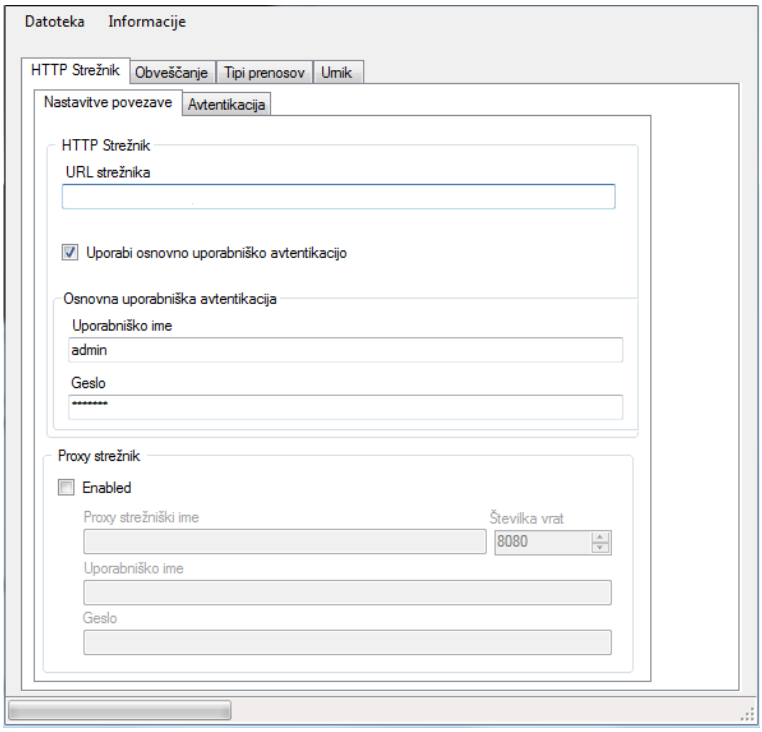
Sledi izvajanje zanke, ki se izvaja glede na interval, ki je minimalno določen za 30 sekund. V vsakem izvajanju te zanke se izvedeta metodi ExecuteDownload in ExecuteUpload. V nadaljevanju je podrobno opisan potek prenosa datotek. Potek prenosa datotek je razviden tudi s slike 4.6, ki je v poglavju Diagram zaporedja spletna storitev.

V ExecuteUpload gre zanka skozi vsa opravila, ki so definirana kot upload. Začetno preverjanje je enako kot pri ExecuteDownload. Razlika je, da se dir izvede nad izvorno lokacijo na odjemalcu. Dobimo seznam datotek, ki jih moramo prenesti. Za vsako datoteko se izvede Upload. V metodi Upload se izvede dir nad strežniškim ciljnim direktorijem. Akcija dir išče imensko enake datoteke. Akcija dir vrne seznam vseh datotek, ki so lahko potencialno prekinjene datoteke, ki se še niso prenesle do konca s pripadajočo kontrolno vsoto in velikostjo datoteke. Za vsako datoteko se preveri, ali se po velikosti in kontrolni vsoti ujema z datoteko na izvorni lokaciji. Če pride do ujemanja, se doda polje range, ki pove, od kod naprej se mora datoteka prenesti. Če ne pride do ujemanja, se datoteka preimenuje. Sledi prenos datoteke na strežnik. Ob koncu pošlje strežnik odgovor, ali se kontrolni vsoti ujemata. V primeru ujemanja se datoteka na strežniku preimenuje in datoteka na odjemalcu se pobriše. V primeru neujemanja se datoteka na strežniku pobriše.

V ExecuteDownload gre zanka skozi vsa opravila, ki so definirana kot download. Strežnik vrne seznam vseh datotek, ki jih je treba prenesti s strežnika na odjemalca. Za vsako datoteko se izvede metoda Download. V metodi Download se najprej preveri, ali je v ciljnem direktoriju na odjemalcu datoteka z enakim imenom. V primeru obstoja enako poimenovane datoteke se izračunata njena kontrolna vsota in velikost. Ta se nato pošlje strežniku, ki preveri, ali sta datoteki enaki. V primeru enakih datotek se v zahtevo doda razpon (range), ki pove, od kod naprej se mora datoteka prenesti. V primeru različnih datotek se bo datoteka prenesla v celoti. Strežnik pošlje datoteko klientu. Na koncu se preverita kontrolni vsoti prejete datoteke in datoteke na strežniku. Če se datoteki ujemata, se s strežniške lokacije datoteka izbriše in vnos se vpiše v bazo. Če kontrolni vsoti nista enaki, se datoteka na odjemalcu pobriše.

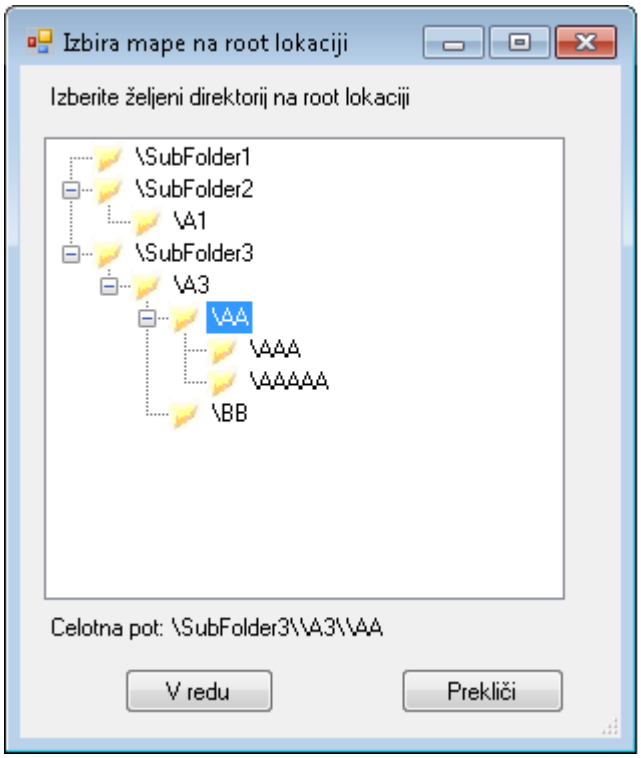
5.3.2 Uporabniški vmesnik

Za uporabniški vmesnik (na sliki 5.5 je prikazan videz uporabniškega vmesnika) je bilo treba implementirati posebno masko za pregledovanje direkto-

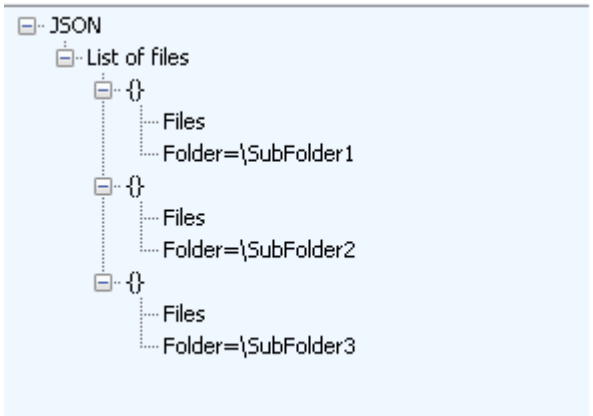


Slika 5.5: Videz uporabniškega vmesnika

rijev na strežniku. Maska je prikazana na sliki 5.6. Ob inicializaciji maske se kliče metoda, ki s pomočjo HttWebRequest proži zahtevo na strežnik. Zahteva ima tip akcije GetFolders. Strežnik odgovori na akcijo s seznamom vseh direktorijev, ki so na dani strežniški lokaciji. Prav tako se ob kliku na posamezni direktorij ponovno sproži zahteva na strežnik, ki vrne seznam direktorijev na izbranem poddirektoriju. Seznam direktorijev se pošlje kot odgovor v obliki JSON. JSON je preprost format za izmenjavo podatkov. Na sliki 5.6 je prikazana maska za pregledovanje direktorijev. JSON format pa je prikazan na sliki 5.7. Sestavljen je iz seznama datotek (Files) na določenem direktoriju (Folder). V primeru, da se datoteke nahajajo v glavnem direktoriju je ime (Folder) prazno. S tem pridobimo celotno direktorijsko strukturo, ki pripada določenemu uporabniku na strežniku.



Slika 5.6: Maska za pregledovanje direktorijev na strežniku



Slika 5.7: JSON odgovor za pridobivanje direktorijske strukture

Poglavje 6

Testiranje

Testiranje je potekalo v dveh fazah. Prva faza je bila generacija testov unit oziroma testiranje modulov. Druga faza je bila priprava testnih postopkov za testiranje programske opreme.

Testni postopki se izvajajo na nameščenem in konfiguriranem sistemu. Med testiranjem se preverja ustreznost namestitve in konfiguracije. Nekatere testne primere je treba izvesti po vsaki namestitvi in spremembi konfiguracije. Testni primeri so namenjeni testiranju mejnih pogojev, vhodnih in izhodnih kontrol. Prav tako se preverja vsebinska in oblikovna pravilnost podatkov. Vsak testni primer je zapisan v lastni tabeli, ki vključuje opis predpogojev, testne korake in kriterije uspešnosti. Testni postopki so ločeni po delih programske rešitve.

Testiranja servisnega modula in spletnih storitev za izmenjavo datotek ni moč testirati individualno, zato ti enoti testiramo skupaj. Skupaj je bilo napisanih okoli 80 testnih postopkov. Primer testnega postopka je prikazan na sliki 6.1.

Vsak testni primer je označen z unikatno številko in imenom. Nato sledi opis predpogojev, ki morajo biti izpolnjeni za uspešno izvedbo testa. Sledi podroben opis testnih korakov. Na koncu so opisani še kriteriji uspešnosti. Testni postopki so bili predani testerju. Vsak testni postopek, ki je bil zavrnjen, se je vpisal v aplikacijo, ki je namenjena testiranju aplikacij.

Št.	Testni primer
2.3	**Ogled podrobnosti o uporabniku**
Predpogoji	
• Delujoča spletna stran • Uspešno izvedeni testi 1.1, 2.1 – 2.2	
Postopek in testni koraki	
1. Odpremo spletno stran 2. Izvedemo prijavo (glej testni primer 1.1) 3. Kliknemo povezavo »Uporabniki« / »Users« 4. Poiščemo v seznamu vseh uporabnikov »Testni Uporabnik Nov« 5. Kliknemo na pripadajočo povezavo »Podrobnosti« / »Details«	
Kriteriji uspešnosti	
• Prikažejo se podrobnosti o Testnem uporabniku.	

Slika 6.1: Primer testnega postopka: Dodajanje novega uporabnika

Pri testiranju modulov preverjamo, ali je posamezni modul pravilno implementiran. Modul je enota kode, ki ima določeno funkcionalnost. Pri objektnem programiranju je modul metoda nekega razreda. Testiranje modulov seveda ne more zagotoviti pravilnega delovanje celotnega produkta, ampak le dele programske opreme, ki pa so neodvisni drug od drugega. Na sliki 6.2 je primer testa unit, ki preverja pravilnost metode GetRolesForUser, ki pridobi vse pravice določenega uporabnika.

```
    /// <summary>
    ///A test for GetRolesForUser, collections should be equal (user roles retrieve from database, user roles retrieve with method getRolesForUser)
    ///</summary>
    [TestMethod()]
    public void GetRolesForUserTest()
    {
        InDocRoleProvider target = new InDocRoleProvider();
        string username = "admin";
        List<string> e = new List<string>();
        string[] actual;
        List<UserRoles> usrroles = new List<UserRoles>(new ServerDataLayerContainer().UserRoles).FindAll(x => x.Users.vcUsername == username);
        foreach (UserRoles item in usrroles)
        {
            e.Add(item.Roles.vcName);
        }
        actual = target.GetRolesForUser(username);
        CollectionAssert.AreEqual(e.ToArray(), actual);
    }
}
```

Slika 6.2: Primer testa unit: Preverjanje pravilnosti metode GetRolesForUser

Poglavje 7

Nadaljnji razvoj

Predlogi za nadaljnji razvoj:

- objava verzij odjemalcev,
- prevzem odjemalca s strežnika (javno dostopno brez prijave v spletno aplikacijo),
- registracija odjemalca na strežniku - strežnik ima evidenco vseh servisnih modulov in njihovega stanja,
- način za avtomatsko posodabljanje odjemalcev (zadnja objavljena verzija je tista, s katero mora biti odjemalec posodobljen).

Z zgoraj naštetimi predlogi bi bil mogoč lažji in enostavnejši pregled nad vsemi odjemalci, ki so postavljeni pri strankah. Z registracijo odjemalca na strežniku bi se vse nastavitve avtomatsko shranjevale na strežnik. Ob morebitnih težavah odjemalcev bi bilo lažje in hitreje najti rešitev. Tako bi lahko le z modifikacijo nastavitev servisnega modula rešili problem. Avtomatsko posodabljanje odjemalcev bi bilo zelo priporočljivo, saj bi v tem primeru prihranili ogromno časa s postavitvijo novih odjemalcev, saj bi vse potekalo preko spleta.

Avtentikacija odjemalca se izvaja v obliki Form in za prijavo zahteva uporabniško ime ter geslo. Smotrno bi bilo implementirati še drugi tip av-

tentikacije - avtentikacija s certifikatom. Avtentikacija s pomočjo certifikatov je namreč varnejša.

Poglavje 8

Sklepne ugotovitve

Cilj diplomskega dela je bil razviti programsko opremo, s katero dosežemo hitro, varno in učinkovito izmenjavo datotek. Glede varnosti je potrebno implementirati še rabo certifikatov, vendar jih zaradi stiske s časom ni bilo še moč implementirati. Glede na to, da je produkt šel uspešno skozi fazo testiranja in se je pojavilo le nekaj manjših hroščev, lahko sklepam, da programska oprema deluje dobro. Seveda pa se bo uspešnost oziroma neuspešnost produkta pokazala, ko bo šel ta v produkcijo.

Pri delu sem se srečala s tehnologijami, ki so bile zame popolnoma nove. Že na začetku sem se morala seznaniti s tehnologijo MVC in s pisanjem servisnih modulov. Največ težav sem imela v fazi implementacije. Težave sta povzročali predvsem dve zahtevi: prenos velikih datotek in implementacija prenosa prekinjenih datotek.

S tem projektom sem pridobila veliko novih izkušenj, saj sem prvič samostojno razvijala produkt. V veliko pomoč mi je bilo pridobljeno znanje s področja razvoja informacijskih sistemov, ki mi je omogočalo lažje in bolj učinkovito izdelavo produkta.

Literatura

[1] A. Adya et al., *Anatomy of the ADO .NET Entity Framework*, International Conference of Managment of Data, 2007, str. 877-888.

[2] A. Freeman, M. MacDonald in M. Szpuszta, *Pro ASP .NET 4.5 in C#*. Apress, 2013.

[3] H. Henrickson, S. R. Hofmann, *IIS 6: The Complete Reference*. McGraw-Hill/Osborne, 2003.

[4] J. Lerman, *Programming Entity Framework*. Manning, 2009

[5] S. Sandersen, A. Freeman, *Pro ASP.NET MVC 3 Framework*. Apress, 2011.

[6] D. Vavpotič, D. Jelen. Prosojnice vaj pri predmetu Razvoj informacijskih sistemov, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Ljubljana, 2010

[7] A. Trolsen, *Pro C# and the .NET 4.5 Framework*, Apress, 2012

[8] R. Žontar, *Zagotavljanje trajnosti v ogrodju .NET*, Diplomsko delo, Fakuleta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, Maribor, 2009.

[9] HTTP Handlers and HTTP Modules Overview. Dostopno na: http://msdn.microsoft.com/en-us/library/bb398986(v=vs.100).aspx

[10] ASP .NET MVC Overview. Dostopno na: http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview

[11] Entity Framework Overview. Dostopno na: http://msdn.microsoft.com/en-us/library/bb399567.aspx

[12] R. Templin (2007) “Introduction to IIS Architecture". Dostopno na: http://www.iis.net/learn/get-started/introduction-to-iis/introduction-to-iis-architecture

[13] Internet Information Services. Dostopno na: http://en.wikipedia.org/wiki/Internet_Information_Services

[14] Introduction to Windows Service Applications. Dostopno no: http://msdn.microsoft.com/en-us/library/aa983650(v=vs.71).aspx

[15] Tutorial – 8 Easy Steps To Create A Web Application With YII – Part 4 And Last. Dostopno na: http://tommasodargenio.com/tutorial-8-easy-steps-to-create-a-web-application-with-yii-part-4-and-last-201.htm

[16] Entity Data Model. Dostopno na: http://msdn.microsoft.com/en-us/library/ee382825.aspx